สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การพัฒนาซอฟแวร์เชิงคอมโพเน็นต์โดยใช้เทคโนโลยีดอทเน็ท

Web Services & Component Software Development Using .Net

T055089

นายจิระเดช ธำรงลักษณ์

นายกรวิชญ์ โตวัฒนกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

เลขหมู่.....
เลขทะเบียน.....55089
วัน,เดือน,ปี.....8 มิ.ย. 2548

b.....
i.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การพัฒนาซอฟแวร์เชิงคอมโพเน็นต์โดยใช้เทคโนโลยีดอทเน็ท

Web Services & Component Software Development Using .Net

โดย

นายจิระเดช ธำรงลักษณ์

นายกรวิชญ์ โตวัฒนกิจ

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา

ดร. ชุติเมษฎ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาซอฟแวร์เชิงคอมโพเน็นต์โดยใช้เทคโนโลยีดอทเน็ท
Web Services & Component Software Development Using .Net

ผู้จัดทำ นายกรวิชญ์ โตวัฒนกิจ รหัสประจำตัว 43010009

นายจิระเดช ธำรงลักษณ์ รหัสประจำตัว 43010068

อาจารย์ที่ปรึกษา
(ดร. วรวัฒน์ ลิ้มโภคา)

อาจารย์ที่ปรึกษา
(ดร.ชุติมยฏ์ ศรีนิลทา)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การพัฒนาซอฟแวร์เชิงคอมโพเน็นต์โดยใช้เทคโนโลยีดอทเน็ท

นายจิระเดช ธำรงลักษณ์
นายกรวิชญ์ โตวัฒนกิจ
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ดร.ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาสถาปัตยกรรม .NET และ J2EE ให้มีความเชื่อถือ และสามารถรองรับการขยายตัวของงานได้มากขึ้น โดยอาศัยหลักการของเทคโนโลยีโหลดบาลานซิ่ง (Load Balancing) เข้ามาช่วยในการแบ่งงานของเซิร์ฟเวอร์และการรองรับความผิดพลาดที่อาจเกิดขึ้น โดยโครงงานนี้ได้รวบรวมทฤษฎี การติดตั้ง และการตั้งค่าต่างๆของสถาปัตยกรรมทั้งสองไว้โดยละเอียด และมีผลการทดสอบประสิทธิภาพของทั้งสองสถาปัตยกรรมเพื่อใช้ในการเปรียบเทียบประสิทธิภาพ

เนื่องจากปัจจุบันมีการใช้อินเตอร์เน็ทเพิ่มขึ้นอย่างแพร่หลาย เว็บเซิร์ฟเวอร์ที่ให้บริการต้องรับภาระหนักจนเกิดการเสียบ่อย ทำให้เสียผลประโยชน์ในทางธุรกิจมากมาย จึงมีวิธีการโหลดบาลานซิ่ง (Load Balancing) เข้ามาช่วยแบ่งเบาภาระจากเดิมที่เซิร์ฟเวอร์ตัวเดียวต้องรับภาระหนักจนเกินไป ก็ทำการเพิ่มเซิร์ฟเวอร์เข้าไปช่วยแบ่งงานกันทำ และยังสามารถเพิ่มความเชื่อถือของเว็บไซน์ได้โดยสามารถรองรับความผิดพลาดที่เกิดขึ้นได้โดยไม่กระทบการทำงานโดยรวม และยังง่ายต่อการดูแลรักษาเพราะสามารถปิดเซิร์ฟเวอร์เพื่อบำรุงรักษาได้

โหลดบาลานซิ่งจึงเป็นอีกทางเลือกในการเพิ่มประสิทธิภาพการทำงานของเว็บแอพพลิเคชันซึ่งสามารถพัฒนาได้ทั้งบนดอทเน็ทเทคโนโลยีและเจทูอีอีเทคโนโลยีซึ่งมีความยากง่ายและข้อดีข้อเสียที่ต่างกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Web Services & Component Software Development Using .Net

Mr. Jiradech Thumrongluck
Mr. Korrawit Towattanakit
Dr. Voravat Limpoka Advisor
Dr. Shutimet Srinilta Advisor
Academic Year 2003

ABSTRACT

This thesis is to analysis, design and implements LoadBalancing technologies in .NET architecture and J2EE architecture that increases reliability and scalability. We gathered all theory, deployment and configuration on both .NET and J2EE technology. Besides, we tested and compared both architectures performance in real system.

Despite the fact that the number of internet users has increased, web servers have to do more hardworking and this is the result that they failed. Business has affected by the failure and lost a lot of money. That is the reason why Load Balancing has developed to share workload between servers and make it more reliable. Moreover, it can prevent failure and easier to maintenance.

Load Balancing is another way to increase performance of web application that can be developed on both .NET and J2EE technology. Besides, they have different strong-point and weak-point.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และความร่วมมือจากหลายๆฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้ปริญญานิพนธ์นี้เสร็จลงได้ คือ ดร.วรวัฒน์ ลิ้มโภคา และ ดร.ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ให้ความเอาใจใส่แนะนำ และความช่วยเหลือเสมอมา ซึ่งต้องขอขอบคุณเป็นอย่างมาก

ขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสทางการศึกษาอย่างเต็มที่ และคอยให้กำลังใจเอาใจใส่เสมอมา ในทุกๆด้าน อันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

นายกรวิชญ์ โตวัฒนกิจ 43010009

นายจิระเดช ธำรงลักษณ์ 43010068

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ(ต่อ)

หน้า

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 แนวคิดและที่มา

เนื่องจากปัจจุบันมีการใช้อินเตอร์เน็ทเพิ่มขึ้นอย่างแพร่หลาย เว็บเซิร์ฟเวอร์ที่ให้บริการต้องรับภาระหนักจนเกิดการเสียบ่อย ทำให้เสียผลประโยชน์ในทางธุรกิจมากมาย จึงมีวิธีการโหลดบาลานซิ่ง (Load Balancing) เข้ามาช่วยแบ่งเบาภาระจากเดิมที่เซิร์ฟเวอร์ตัวเดียวต้องรับภาระหนักจนเกินไป ก็ทำการเพิ่มเซิร์ฟเวอร์เข้าไปช่วยแบ่งงานกันทำ และยังสามารถเพิ่มความเชื่อถือของเว็บไซน์ได้โดยสามารถรองรับความผิดพลาดที่เกิดขึ้นได้โดยไม่กระทบการทำงานโดยรวม และยังง่ายต่อการดูแลรักษาเพราะสามารถปิดเซิร์ฟเวอร์เพื่อบำรุงรักษาได้

โหลดบาลานซิ่งจึงเป็นอีกทางเลือกในการเพิ่มประสิทธิภาพการทำงานของเว็บแอปพลิเคชัน ซึ่งสามารถพัฒนาได้ทั้งบนดอทเน็ทเทคโนโลยีและเจทูอีอีเทคโนโลยีซึ่งมีความยากง่ายและข้อดีข้อเสียที่ต่างกัน

1.2 วัตถุประสงค์

1. เพื่อศึกษาเทคโนโลยี .Net โครงสร้างและการทำงานของ .Net ว่ามีลักษณะอย่างไร สามารถนำมาใช้ประโยชน์ได้อย่างไร
2. ศึกษาการทำงานของเน็ทเวิร์คโหลดบาลานซิ่ง(Network Load Balancing) บน Windows 2000 Advance Server
3. เพื่อความรู้ที่ได้จากการใช้เน็ทเวิร์คโหลดบาลานซิ่งนำมาประยุกต์ใช้กับแอพพลิเคชั่นอื่นๆ ต่อไป
4. เพื่อศึกษาเทคโนโลยี J2EE โครงสร้างและการทำงานของ J2EE ว่ามีลักษณะอย่างไร สามารถนำมาใช้ประโยชน์ได้อย่างไรและจะทำการเพิ่มประสิทธิภาพการทำงานด้วยวิธีโหลดบาลานซิ่งได้อย่างไร
5. เพื่อศึกษาการทำงานของโหลดบาลานซิ่งบน BEA WebLogic 7.0
6. เปรียบเทียบประสิทธิภาพการทำงานของ BEA WebLogic 7.0 และ IIS Server
7. เปรียบเทียบประสิทธิภาพการทำโหลดบาลานซิ่งของ BEA WebLogic และไมโครซอฟท์

1.3 ผลที่คาดว่าจะได้รับ

1. ได้รับความรู้ความสามารถในเทคโนโลยีของทั้งดอทเน็ทและเจทูอีอี
2. ได้รับความรู้ความสามารถในการออกแบบสถาปัตยกรรมดอทเน็ทและเจทูอีอี
3. ได้รับความรู้ความสามารถในในการติดตั้งเซิร์ฟเวอร์แบบโหลบาลานซิ่งทั้งดอทเน็ทและเจทูอีอี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. ได้รับความรู้ความสามารถในด้านน็ทเวิร์คที่ต้องติดตั้งเซิร์ฟเวอร์

1.4 ขอบเขตของการพัฒนา

โครงการนี้มุ่งเน้นไปที่การเพิ่มความสามารถของเซิร์ฟเวอร์ที่มีอยู่โดยใช้วิธีโหลดบาลานซิ่ง การออกแบบสถาปัตยกรรมที่เหมาะสมกับแต่ละเทคโนโลยี และการทดสอบประสิทธิภาพการทำงานของสถาปัตยกรรม .NET และ J2EE รวมถึงการประยุกต์เพื่อนำไปใช้กับแอพพลิเคชันจริง

1.5 วิธีการดำเนินงาน

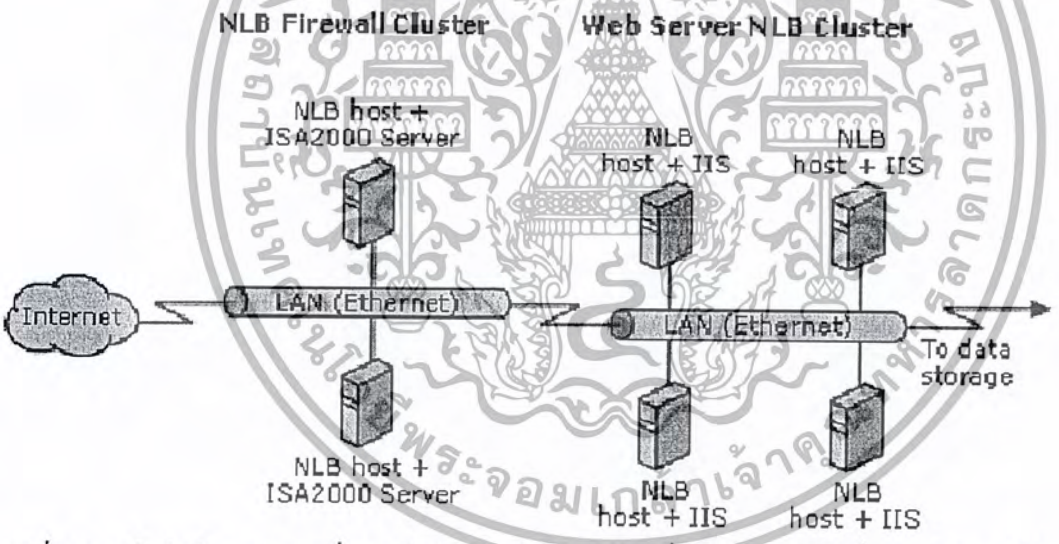
การทำโครงงานเริ่มที่การศึกษาข้อมูลของเทคโนโลยีดอทเน็ทและเจทูอีอีก่อนเมื่อเข้าใจแล้วจึงศึกษาข้อมูลของโหลดบาลานซิ่งโดยเริ่มที่เทคโนโลยีของไมโครซอฟท์ก่อน โดยอีกคนศึกษาเทคโนโลยีโหลดบาลานซิ่งของเว็บโลจิก จากนั้นก็ทำการติดตั้งทดสอบการทำงานโดยเขียนเว็บขึ้นมาเองแบ่งเป็นสะแตติกเว็บ (static web) และ ไดนามิกเว็บ (dynamic web) โดยทดสอบการแบ่งงานก่อนว่าถูกต้องหรือไม่ แล้วตึงทดสอบโดยวิธีการ Stress Test โดยการส่งคำร้องขอใช้งานเว็บจำนวนมากเพื่อเก็บผลแล้วนำมาวิเคราะห์เปรียบเทียบประสิทธิภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีและความรู้พื้นฐานเกี่ยวกับ Load Balancing บน Microsoft Windows 2000 Advanced Server

2.1 แนะนำโหลดบาลานซิ่งของไมโครซอฟท์

บริการเนตเวิร์คโหลดบาลานซิ่งสามารถเพิ่มความสามารถด้านคุณภาพ และขนาดของโปรแกรมที่บริการอินเตอร์เน็ตยกตัวอย่างเช่น เซิร์ฟเวอร์ที่ให้บริการด้านเว็บ ด้านเอฟทีพี(FTP), พร็อกซี่, วีพีเอ็นหรือใช้กับบริการอื่นๆที่จะเกิดการเสี่ยงต่อความผิดพลาด ในการใช้คอมพิวเตอร์เครื่องเดียวเราสามารถรองรับข้อมูลได้ในอัตราที่จำกัด วิธีโหลดบาลานซิ่งก็คือการนำเครื่องคอมพิวเตอร์ตั้งแต่ 2 เครื่องขึ้นไปมาเชื่อมกันโดยใช้โปรแกรมเดียวกัน โดยทำงานเสมือนเป็นเครื่องเดียวโหลดบาลานซิ่งสามารถเพิ่มความน่าเชื่อถือและประสิทธิภาพให้กับserver ที่บริการเว็บ หรือเซิร์ฟเวอร์ ที่บริการแอพพลิเคชันที่มีความเสี่ยงต่อการล้มเหลว



รูปที่ 2-1 เครือข่ายโหลดบาลานซิ่ง 2 เครือข่าย ต่อเข้าด้วยกัน โดยที่ระบบแรกจะใช้เครื่องคอมพิวเตอร์ร่วมกัน 2 เครื่อง ส่วนระบบที่ 2 จะใช้เครื่องคอมพิวเตอร์ร่วมกัน 4 เครื่อง

แต่ละเครื่องจะทำงานโดยการคัดลอกแอพพลิเคชั่นของเซิร์ฟเวอร์มาไว้ที่เครื่องของตนเอง ยกตัวอย่างเช่น เว็บเซิร์ฟเวอร์ เอฟทีพี และเทลเน็ต เซิร์ฟเวอร์ (Telnet Server) เป็นโหลดบาลานซิ่ง จะแบ่งคำขอของเครื่องไคลเอนท์ไปให้เครื่องต่าง ๆ ภายในคลัสเตอร์ ส่วนเรื่องที่ว่าจะให้เครื่องไหนรับคำขอขนาดไหนนั้นขึ้นอยู่กับการคอนฟิกของผู้ใช้ ผู้ใช้สามารถเพิ่มเครื่องคอมพิวเตอร์เข้าไปในคลัสเตอร์ได้ เพื่อเพิ่มโหลดให้สามารถรองรับได้เพิ่มขึ้น โหลดบาลานซิ่งสามารถบอกให้ข้อมูลทั้งหมดไปทำงานที่เครื่องเดียวเลยก็ได้ ซึ่งเราเรียกว่า ดีฟอลท์โฮส (default host)

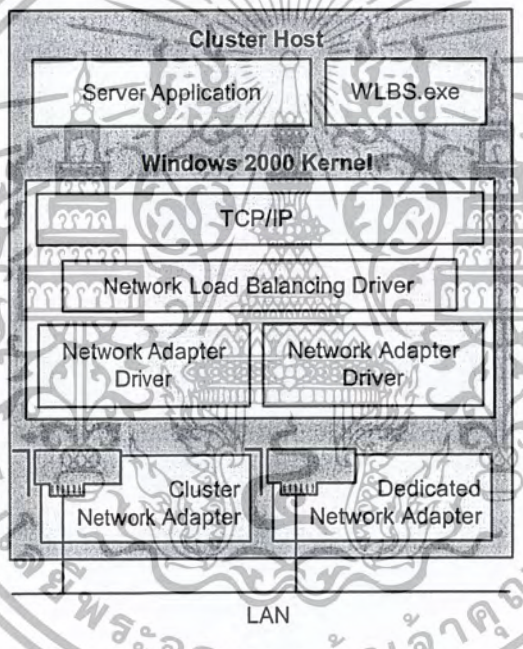
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โหลดบาลานซิ่ง อนุญาตให้เครื่องที่อยู่ในคลัสเตอร์ใช้แอดเดรสเดียวกันซึ่งเราเรียกว่า คลัสเตอร์ไอพี แอดเดรส (Cluster IP Address)

สำหรับ แอพพลิเคชันที่ใช้โหลดบาลานซิ่งมื่อเครื่องใดเครื่องหนึ่งเกิดล้มเหลว(fail) หรือปิดเครื่องไป เครื่องที่เหลือจะแบ่งการทำงานทั้งหมดอีกครั้งคือเอาการทำงานของเครื่องที่ล้มเหลวไปใช้กับเครื่องอื่นที่ยังทำงานได้เพื่อให้ทำงานต่อไปได้

2.2 วิธีใช้งานโดยรวมของเน็ตเวิร์คโหลดบาลานซิ่ง

เน็ตเวิร์คโหลดบาลานซิ่งทำงานบนเครือข่ายของวินโดว์ และทำงานบน TCP/IP รูปด้านล่างแสดงถึงความสัมพันธ์ระหว่างเน็ตเวิร์คโหลดบาลานซิ่งและคอมโพเน็นท์ของซอฟท์แวร์ที่ทำงานบน เน็ตเวิร์คโหลดบาลานซิ่ง



รูปที่ 2-2 ความสัมพันธ์ระหว่างเน็ตเวิร์คโหลดบาลานซิ่งและ คอมโพเน็นท์ของซอฟท์แวร์ที่ทำงานบน เน็ตเวิร์คโหลดบาลานซิ่ง

บางแอพพลิเคชั่นต้องทำงานเกี่ยวกับฐานข้อมูลโดยการเปลี่ยนแปลงจากไคลเอนท์ เมื่อแอพพลิเคชั่นนั้นถูกทำโหลดบาลานซิ่งในคลัสเตอร์ การเปลี่ยนแปลงต้องเปลี่ยนแปลงแบบไม่มีผลกระทบต่อสิ่งแวดล้อมรอบข้าง โดยที่แต่ละเครื่องจะมีฐานข้อมูลเป็นของแต่ละเครื่องเลยและจะนำมารวมกันเมื่อจำเป็น อีกทางเลือกนึงก็คือเครื่องที่เป็นเครื่องกลางแชร์ข้อมูลของฐานข้อมูล เมื่อมีการเปลี่ยนแปลงก็เปลี่ยนแปลงเลย ยกตัวอย่างเช่น เว็บไซต์หนึ่งสามารถมีอยู่ในเครื่องทุกเครื่องได้เพื่อเพิ่มความเร็วและป้องกันความผิดพลาด อย่างไรก็ตาม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฐานข้อมูลจะร้องขอให้มีการส่งต่อข้อมูลไปที่ฐานข้อมูลที่เป็นศูนย์กลางเพื่อที่จะนำไปเปลี่ยนแปลงทุกๆเวบไซต์ที่เกี่ยวข้อง

2.3 หลักการทำงานของเน็ตเวิร์คโหลดบาลานซิ่ง

เน็ตเวิร์คโหลดบาลานซิ่งจะเพิ่มประสิทธิภาพและขนาดขององค์กรโดยการใช้คลัสเตอร์จากเครื่องคอมพิวเตอร์ตั้งแต่ 2 เครื่องขึ้นไปมาทำงานด้วยกัน เครื่องไคลเอนท์ที่เชื่อมอินเตอร์เน็ตอยู่จะไม่รู้เลยว่ามีหลายเครื่องทำงานช่วยกันอยู่ แต่จะเห็นเป็นเพียงไอพีแอดเดรสเดียวปกติ เครื่องที่ให้บริการก็ไม่จำเป็นต้องระบุว่าพวกมันกำลังทำงานอยู่ในระบบคลัสเตอร์ อย่างไรก็ตาม กลุ่มเครื่องคอมพิวเตอร์ที่ทำงานแบบโหลดบาลานซิ่งมีข้อแตกต่างจากเครื่องคอมพิวเตอร์ที่เปิดให้บริการเพียงเครื่องเดียว ก็คือ เครื่องที่ทำงานหลายเครื่องจะช่วยลดปัญหาการรบกวนถึงแม้ว่าจะมีเครื่องใดเสียก็ตาม และจะให้บริการข้อมูลได้ดีกว่าแบบที่เครื่องเดียวเป็นเซิร์ฟเวอร์

การทำเน็ตเวิร์คโหลดบาลานซิ่งสามารถเพิ่มประสิทธิภาพได้โดยการเปลี่ยนเส้นทางข้อมูลที่เข้ามาในเครือข่ายหากเครื่องที่ได้รับนั้นให้บริการไม่ได้ ข้อมูลนั้นจะเปลี่ยนทางไปที่เครื่องอื่นที่ยังทำงานได้อยู่ และทำให้สามารถรับข้อมูลที่มาจากเครื่องไคลเอนท์ได้มากขึ้น

เน็ตเวิร์คโหลดบาลานซิ่งรองรับการทำงานในปริมาณมากได้โดยการกระจายภาระของเน็ทเวิร์คที่เข้ามาไปยังหนึ่งหรือสอง virtual IP address(cluster IP address) ไปยังโหลดที่รวมไว้ ตัวโฮสในนั้นจะตอบไปยังผู้ขอใช้ต่างๆในเวลาเดียวกันแม้จะมีการร้องขอเข้ามาหลายๆครั้งจากผู้ขอคนเดียว เช่น เวบเบราเซอร์อาจจะมีรูปภาพหลายๆรูปในหน้าเว็บเพียงหน้าเดียวที่มาจากโฮสที่ต่างกันในเน็ตเวิร์คโหลดบาลานซิ่งสิ่งนี้ทำให้การทำงานและการตอบกลับไปยังผู้ขอใช้เร็วขึ้น

เน็ตเวิร์คโหลดบาลานซิ่งทำให้ทุกโฮสบนซับเน็ตหนึ่งๆ ตรวจพบการเข้ามาของเน็ทเวิร์คสำหรับคลัสเตอร์ไอพีแอดเดรสพร้อมๆกัน ในแต่ละโฮสเน็ตเวิร์คโหลดบาลานซิ่งไดรเวอร์(Network Load Balancing driver) ทำตัวบอกเหมือนตัวกรองระหว่างคลัสเตอร์อะแดปเตอร์ไดรเวอร์(cluster adapter driver) และ TCP/IP stack เพื่อที่จะกระจายความคับคั่งของเน็ตเวิร์คบนโฮส

เน็ตเวิร์คโหลดบาลานซิ่งใช้หลักการทำงานแบบกระจาย(distributed algorithm) เพื่อจับคู่ผู้ใช้งานกับโฮสแบบสถิติบนไอพีแอดเดรส และพอร์ต ของผู้ใช้งาน การติดต่อระหว่างโฮสจะไม่จำเป็นสำหรับการทำงานนี้ ขณะกำลังตรวจสอบแพ็กเกจที่เข้ามา ทุกโฮสจะทำการจับคู่พร้อมๆกันเพื่อจะได้รู้ว่าโฮสไหนควรจัดการกับแพ็กเกจนั้นๆโดยเร็วที่สุด การจับคู่จะไม่เปลี่ยนแปลงถ้าจำนวนโฮสไม่เปลี่ยนแปลง

2.4 การคอนเวอร์เจนซ์(Convergence)

เพื่อที่จะทำงานไปพร้อมๆกัน โฮสต่างของเน็ตเวิร์คโหลดบาลานซิ่ง จะแลกเปลี่ยนสัญญาณฮาทบีท(ข้อความที่โฮสในเน็ตเวิร์คโหลดบาลานซิ่งส่งไปยังอีกโฮสเพื่อตรวจสอบการสื่อสารที่ผิดพลาด) ในนั้น ไอพีมัลติแคสท์(IP multicasting)อนุญาตให้โฮสดูสถานะของคลัสเตอร์ เมื่อสถานะของคลัสเตอร์ เปลี่ยน เช่น โฮสเกิดข้อผิดพลาด เอาโฮสออก หรือ เพิ่มเข้ามาในคลัสเตอร์เน็ตเวิร์คโหลดบาลานซิ่ง(cluster Network Load Balancing)จะมาทำขบวนการที่เรียกว่า “ คอนเวอร์เจนซ์ (Convergence) ” เพื่อให้โฮสแลกเปลี่ยน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อความที่ถูกจำกัดจำนวนไว้ ทำการตรวจดูสถานะใหม่ สถานะเดิม และตัดสินใจเลือกโฮสที่มีค่าลำดับความสำคัญ(priority)สูงสุดเป็นดีฟอลท์โฮสต์(default host) ตัวใหม่ เมื่อทุกโฮสมีความเห็นเอกฉันท์ในสถานะใหม่ของคลัสเตอร์ พวกมันก็จะบันทึกลงอีเวนท์ล็อก(event log)ของวินโดวส์ ขบวนการนี้กินเวลาน้อยกว่า 10 วินาที

ระหว่างการคอนเวอร์เจนซ์ (Convergence) โฮสที่เหลือก็ทำการจัดการกับข้อมูลจากเน็ตเวิร์ค ที่เข้ามาต่อ เมื่อทำคอนเวอร์เจนซ์ (Convergence) เสร็จเส้นทางที่ไปยังโฮสที่ทำงานผิดพลาดจะถูกส่งใหม่ไปยังโฮสที่เหลือ การทำให้ทุกโหลดทำงานเท่าๆกันคือการแบ่งส่วนใหม่ของทีซีพี และยูดีพีพอร์ตระหว่างโฮสที่เหลือให้เท่ากัน

ถ้าโฮสถูกเพิ่มคอนเวอร์เจนซ์ (Convergence) อนุญาตให้โฮสนี้รับส่วนแบ่งการทำงานที่เท่าๆกัน การขยายนี้ไม่กระทบการทำงานทั้งหมดและจะถูกติดตั้งโดยทั้งอินเตอร์เน็ตไกลเอนท์และเซิร์ฟเวอร์ไม่เห็นซึ่งกันและกัน อย่างไรก็ตามมันอาจกระทบส่วนของผู้ขอใช้ที่ทำการต่อแบบทีซีพี เมื่อการทำงานแบบ Affinity (การส่งคำร้องขอที่มาจากไอพีแอดเดรสเดียวกันไปยังโฮสตัวเดียวกัน) ถูกเลือกมาใช้ เพราะผู้ขอใช้อาจถูกจับคู่ให้ทำงานกับโฮสต่างๆกัน

เน็ตเวิร์คโหลดบาลานซิ่งจะสมมุติว่ามีการทำงานปกตินานเท่าที่มีการแลกเปลี่ยนฮาทบีท(Heartbeats) ระหว่างโฮส ถ้าโฮสอื่นไม่ได้รับการตอบกลับจากหลายๆการแลกเปลี่ยน พวกมันเริ่มการคอนเวอร์เจนซ์ (Convergence) เพื่อทำการกระจายโหลดไปให้โฮสอื่น

คุณสามารถควบคุมระยะเวลาการแลกเปลี่ยนข้อความและจำนวนที่สูญหายของการคอนเวอร์เจนซ์ (Convergence) ได้ ค่าปกติคือ 1วินาทีและ 5 ข้อความ จะสามารถเปลี่ยนแปลงได้ในรีจิสตรีถ้าจำเป็น

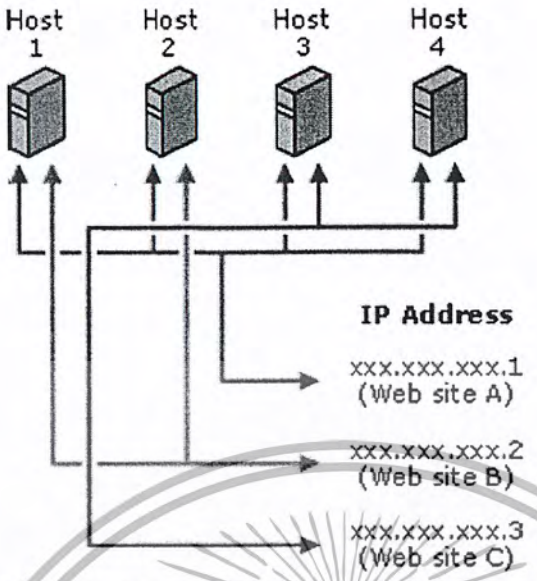
2.5 เวอร์ชวล คลัสเตอร์(Virtual Cluster)

ในปัจจุบันมีการเพิ่มขึ้นของการมีหลายแอพพลิเคชั่น หรือเว็บไซต์ บนเน็ตเวิร์คโหลดบาลานซิ่งคลัสเตอร์(Network Load Balancing (NLB) cluster) การทำเช่นนั้นต้องมีการกำหนดกฎซึ่งเรียกว่า "การกำหนดพอร์ต (Port rules)" มาใช้กับแต่ละแอพพลิเคชั่น หรือเว็บไซต์

ใน Window Server 2003 คุณสามารถกำหนดหลายๆเน็ตเวิร์คโหลดบาลานซิ่งคลัสเตอร์ (Network Load Balancing clusters) บนอุปกรณ์เน็ตเวิร์คตัวเดียวกันและทำการกำหนดพอร์ต (Port rules) สำหรับแต่ละไอพีแอดเดรสสิ่งเหล่านี้เรียก "เวอร์ชวล คลัสเตอร์ (Virtual Clusters)"

คุณสามารถใช้เวอร์ชวล คลัสเตอร์ (Virtual Clusters) กันการไหลของเน็ตเวิร์คเข้าแต่ละโฮสสำหรับแต่ละแอพพลิเคชัน โดยไม่กระทบแอพพลิเคชัน อื่นบนโฮสนั้น คุณสามารถใช้เวอร์ชวล คลัสเตอร์ (Virtual Clusters) กำหนดแต่ละแอพพลิเคชัน, เว็บไซต์ หรือเวอร์ชวล ไอพีแอดเดรส (Virtual IP address) เพื่อกำหนดตัวคอมพิวเตอร์ที่จะอยู่ในคลัสเตอร์ของคุณ ดังรูปข่างล่างนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2-3 การทำงานของ เวอร์ชวล คลัสเตอร์ (Virtual Clusters)

ในรูปนี้จะมี 4 เน็ตเวิร์คโหลดบาลานซิ่งโฮสผ่านทาง ของ เวอร์ชวล คลัสเตอร์ (Virtual Clusters) และไอพีแอดเดรส การเข้าออก เน็ตเวิร์ค มีดังนี้

-เข้าเว็บ A (IP address nnn.nnn.nnn.1) จะถูกส่งไปยังทั้ง 4 โฮสนี้ตัวนี้

-เข้าเว็บ B (IP address nnn.nnn.nnn.2) ซึ่งเป็น virtual cluster จะถูกส่งไปยังโฮสต์ 1 และ 2

-เข้าเว็บ B (IP address nnn.nnn.nnn.3) ซึ่งเป็น virtual cluster จะถูกส่งไปยังโฮสต์ 3 และ 4

2.6 ทฤษฎีการทำงานของโหลดบาลานซิ่ง(Load Balancing Algorithm)

เมื่อมีแพ็กเกจมาถึง ทุกโฮสจะทำการเก็บค่าสถิติ(Statistical Mapping) เพื่อหาว่าโฮสใดควรจะจัดการกับแพ็กเกจนั้น การแมพ(mapping)นั้นใช้ฟังก์ชันแบบสุ่ม (randomization function) ที่คำนวณระดับความสำคัญ(priority) ของโฮสที่อ้างอิงกับไอพีแอดเดรส, พอร์ต และข้อมูลอื่นๆ เพื่อให้การโหลดบาลานซ์ดีที่สุด โฮสที่ถูกเลือกโดยดีฟอลท์ โฮสต์(default host) จะส่งแพ็กเกจต่อไปยังเน็ตเวิร์คบน TCP/IP และโฮสอื่นจะไม่สนใจแพ็กเกจนั้น การแมพ(mapping) จะไม่เปลี่ยนถ้าสมาชิกในกลุ่มของโฮสไม่เปลี่ยน เพราะฉะนั้นมั่นใจได้ว่าไอพีแอดเดรสและพอร์ตที่เข้ามาจะถูกแมพ เข้ากับโฮสเดิมตลอด อย่างไรก็ตามถ้ามีการเปลี่ยนแปลงของโฮสในกลุ่มไอพีแอดเดรสและพอร์ต นั้นๆจะไม่สามารถ ระบุได้ว่าจะเป็นโฮสตัวเดิม เนื่องจากจะมีการใช้ฟังก์ชันสุ่มใหม่มาคำนวณโดยใช้โฮสในกลุ่มเก่าและใหม่มาคำนวณเพื่อลดการแมพ(mapping) ใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7 ไอพีแอดเดรส

ขณะติดตั้งเน็ตเวิร์คโหลดบาลานซิ่ง จะให้ใส่ค่าไอพีแอดเดรสหลัก(primary IP address) ซึ่งจะแทนเวอร์ชวลไอพีแอดเดรส(Virtual IP address)ที่ทุกโฮสจะต้องมี แต่ละโฮสยังต้องมีไอพีเครื่อง(Dedicated IP address)สำหรับการติดต่อถึงแต่ละโฮสโดยตรงเน็ตเวิร์คโหลดบาลานซิ่ง จะไม่บาลานซ์การส่งข้อมูลของ ไอพีเครื่อง(dedicated IP address)แต่จะบาลานซ์ การส่งข้อมูลจากทุกไอพีแอดเดรสยกเว้นไอพีที่เป็นไอพีเครื่อง (Dedicated IP address)

2.8 ลำดับความสำคัญของแต่ละโฮสต์ (Host Priorities)

ทุกโฮสจะมีลำดับความสำคัญ(host priority) ไม่ซ้ำกันระหว่าง 1 ถึง 32 โดยเลขที่ต่ำกว่าบอกถึงความสำคัญ (priority) ที่สูง โฮสที่มีความสำคัญ(priority) สูงสุดจะเรียกว่า "ดีฟอลท์โฮสต์(default host)" มันเป็นตัวจัดการทุกข้อมูลจากผู้ใช้ที่เข้ามายังเวอร์ชวลไอพีแอดเดรส(virtual IP address)ที่ยังไม่ถูกโหลดบาลานซิ่งถ้าโฮสนี้เกิดเสียโฮสที่มีความสำคัญ(priority)สูงสุดถัดมาจะเป็น ดีฟอลท์โฮสต์(default host)

2.9 การกำหนดพอร์ตการเชื่อมต่อ(Port Rules)

เน็ตเวิร์คโหลดบาลานซิ่งใช้ การกำหนดพอร์ต(port rules)เพื่อปรับแต่งโหลดบาลานซิ่งสำหรับการติดต่อพอร์ต ของเครื่องเซิร์ฟเวอร์ การกำหนดพอร์ต(port rules) สามารถเลือกได้ 2 แบบ คือแบบที่มีหลายโฮสต์(multiple-host) และแบบโฮสต์เดียว(single-host)โดยแบบที่เป็นหลายโฮสต์(multiple-host)เมื่อมีข้อมูลเข้ามาจะถูกแบ่งระหว่างทุกโฮส และ เปอร์เซ็นต์ของโหลด(load percentage) สามารถกำหนดลงไปในแต่ละโฮสได้ เปอร์เซ็นต์ของโหลดอนุญาตให้โฮสที่มีส่วนแบ่งงานสูงรับส่วนงานมากตามอัตราส่วนแบบที่เป็นโฮสต์เดียว(single-host)จะส่งข้อมูลทั้งหมดไปยังโฮสที่มีค่าลำดับความสำคัญ(handling priority)สูงสุดและค่าลำดับความสำคัญสามารถแบ่งการทำงานของแต่ละโฮสให้แยกกันได้โดยกำหนดพอร์ตที่เซิร์ฟเวอร์และสามารถกำหนดงานเฉพาะที่จะทำได้ การกำหนดพอร์ตยังสามารถป้องกันแพ็กเกจที่เข้ามาในพอร์ต ที่เราไม่ต้องการได้

เมื่อใช้การกำหนดพอร์ตแบบที่มีโฮสต์หลายตัว(multiple-host) จะต้องเลือกหนึ่งในสามของค่าอัฟฟินิตี้(Client Affinity Mode) เมื่อไม่มีการใช้ค่าอัฟฟินิตี้(Affinity) โหลดบาลานซิ่งจะบาลานซ์ข้อมูลจากหนึ่งไอพีแอดเดรส ใดๆและพอร์ต ที่แตกต่างกันกับแบบที่มีโฮสต์หลายตัว(multiple-host) กล่าวคือเมื่อไคลเอนท์ส่งข้อมูลมาหลายครั้งก็มีโอกาสที่จะติดต่อกับโฮสใหม่ได้ ในการที่จะจัดการผู้ร้องขอใดๆให้ติดต่อกับโฮสใดโฮสหนึ่งจึงมีซิงเกิลไคลเอนท์(single-client) ซึ่งถูกกำหนดไว้เป็นดีฟอลท์ มาจัดการโดยบาลานซ์ทุกข้อมูลที่เข้ามาจากไอพีแอดเดรสส่วนรูปแบบสุดท้ายคลาส ซี(class C affinity)จะบาลานซ์เฉพาะไอพีแอดเดรส ที่เป็นคลาส ซี(class C)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.10 การแบ่งงานระหว่างเซิร์ฟเวอร์

เน็ตเวิร์คโหลดบาลานซิ่งใช้เลเยอร์ทูบรอดแคสท์(layer-two broadcast) หรือมัลติแคสท์(multicast) ในการแจกจ่ายข้อมูลที่เข้ามาไปยังทุกๆโฮสพร้อมกัน ในค่าดีฟอลท์จะเลือกไว้ที่ยูนิแคสท์(unicast)โหมดโหลดบาลานซิ่งจะได้แมคแอดเดรส(MAC Address) ใหม่เพื่อใช้ทำการเชื่อมต่อคลัสเตอร์(cluster adapter) และทุกโฮสจะมีแมคแอดเดรสเหมือนกัน ทั้งเครื่องที่เป็นเครื่องปกติ(dedicated IP address) และเครื่องที่เป็นคลัสเตอร์(cluster IP address) จะมีแมคแอดเดรสเดียวกัน เพราะฉะนั้นการติดต่อระหว่างโฮสจึงทำไม่ได้ ทุกข้อมูลที่เข้ามาทุกโฮสจะได้รับและส่งไปยังเน็ตเวิร์คโหลดบาลานซิ่งไดรเวอร์(Network Load Balancing driver)เพื่อกรองข้อมูลอีกที แมคแอดเดรส ได้มาจากไอพีเสมือน (Virtual IP address) ที่ทุกโฮสต้องมีเหมือนกัน เช่น ถ้าไอพีแอดเดรสเป็น 1.2.3.4 ยูนิแคสต์ แมคแอดเดรส (unicast MAC address) จะเป็น 02-BF-1-2-3-4 โหลดบาลานซิ่ง จะแปลงค่าแมคแอดเดรสโดยอัตโนมัติ โดยการตั้งค่าในรีจิสตรี(registry)และโหลดไดรเวอร์(adapter's driver)ใหม่โดยที่ไม่ต้องเริ่ม การทำงานของระบบปฏิบัติการ

ถ้าโฮสต่างๆต่อเข้ากับสวิทช์ แทนที่จะเป็นฮับ การใช้แมคแอดเดรสก็จะสับสนเพราะสวิทช์ต้องการแมค แอดเดรส(MAC address) ที่มีได้เพียงตัวเดียวในแต่ละพอร์ต เพื่อหลีกเลี่ยงปัญหา จึงมีการเปลี่ยนแมคแอดเดรส จาก 02-BF-1-2-3-4 เป็น 02-h-1-2-3-4 เวลาส่งแพ็กเกจออกไป เมื่อ h คือค่าความสำคัญ(priority) ของโฮส การทำเช่นนี้ป้องกันไม่ให้สวิทช์ จำแมคแอดเดรส จริงๆของโฮสได้ และเมื่อมีแพ็กเกจ เข้ามาก็จะส่งไปทุกพอร์ตที่โฮสต่ออยู่ ถ้าโฮสต่ออยู่กับฮับ การแปลงแมคแอดเดรส ของ ยูนิแคสท์(unicast) ไม่จำเป็นต้องทำก็ได้เพื่อหลีกเลี่ยงการท่วมของข้อมูลบน สวิทช์ที่อยู่ข้างบนฮับ สามารถทำได้โดยแก้ค่า MaskSourceMAC เป็น 0 ใน registry

การใช้ยูนิแคสท์(unicast) โหมดมีข้อเสียตรงที่การติดต่อระหว่างโฮสเนื่องจากแพ็กเกจ ที่ส่งไปยังโฮสอื่นมีแมคแอดเดรส เดียวกัน สามารถแก้ได้โดยการเพิ่มอุปกรณ์เชื่อมต่อเน็ตเวิร์ค(network adapter) ตัวที่สองในทุกโฮสโดยใช้ซับเน็ตภายใน(local subnet)ในการติดต่อระหว่างโฮสและ ดาต้าเบสเซิร์ฟเวอร์ (database server)

โหมดที่สองได้แก่มัลติแคสท์(multicast) โหมด จะให้ค่าแมคแอดเดรส เป็น มัลติแคสท์แอดเดรส(multicast address) เช่นไอพีเอดเดรสเป็น 1.2.3.4 จะมีมัลติแคสท์แมคแอดเดรส(multicast MAC address)เป็น 03-BF-1-2-3-4 เนื่องจากทุกโฮสยังคงมีแมคแอดเดรส เฉพาะของแต่ละโฮสเหมือนเดิม (ไอพีกลาง เป็นมัลติแคสท์แมคแอดเดรส และไอพีเครื่อง เป็นแมคแอดเดรส ของมันเอง) โหมดลดความต้องการอุปกรณ์เน็ตเวิร์คตัวที่สองสำหรับการติดต่อระหว่างโฮสลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3

ทฤษฎีและความรู้พื้นฐานเกี่ยวกับ Load Balancing บน BEA WebLogic 7.0

<u>แนะนำเว็บโลจิกโหลดบาลานซิ่ง (WebLogic Load Balancing)</u>

ทุกๆ WebLogic เซิร์ฟเวอร์ ใน คลัสเตอร์ ต้องการถูกใช้งานอย่างเต็มที่เพื่อรองรับงานปริมาณมาก จึงมีการใช้วิธีโหลดบาลานซิ่งมาใช้ในการแบ่งงานระหว่างโฮส ถ้าโฮสไหนเกิดเสียโหลดบาลานซิ่งจะรับประกันว่ามีโฮสอื่นมารองรับการทำงานแทนโฮสนั้นได้ เราสามารถตั้งค่าโหลดบาลานซิ่งต่างๆได้เมื่อติดตั้ง คลัสเตอร์ โดยจะมีวิธีการทำได้ 3 แบบดังนี้

1. Round-robin(default)
2. Weight based
3. Random

Round-Robin โหลดบาลานซิ่ง

Round-Robin จะทำการวนโฮสใน คลัสเตอร์ ไปเรื่อยในการทำงาน ตัวโฮสที่ทำคลัสเตอร์ จะรับข้อมูลมาแล้วส่งไปยังเซิร์ฟเวอร์ ที่ว่างตามลำดับการเข้ามาของข้อมูลวนไปเรื่อยๆ การใช้วิธีนี้เหมาะกับเครื่องที่มีประสิทธิภาพพอๆกัน

การทำ Round-Robin นั้นง่าย ประหยัด และสามารถคาดการได้ แต่ข้อเสียหลักของมันจะเกิดเมื่อมีเซิร์ฟเวอร์ หนึ่งช้ากว่าตัวอื่นจะทำให้ข้อมูลที่เข้ามาต้องรอเซิร์ฟเวอร์ นั้นทำให้ขบวนการทั้งหมดช้าลงได้

Weight-Based โหลดบาลานซิ่ง

ดีกว่า Round-robin โดยที่ weight-based จะนำค่าน้ำหนักของแต่ละ เซิร์ฟเวอร์ มาคิดด้วย เซิร์ฟเวอร์แต่ละตัวจะมีค่าน้ำหนักตั้งแต่ 1-100 โดยอัตราส่วนนี้จะขึ้นกับ เซิร์ฟเวอร์ อื่นด้วย เช่น มี เซิร์ฟเวอร์ เครื่องหนึ่งตั้งค่าไว้ที่ 50 นอกนั้นเป็น 100 หมด เซิร์ฟเวอร์ ตัวนั้นก็จะแบ่งงานไป 50 ของอัตราส่วนน้ำหนักของเซิร์ฟเวอร์ ที่เหลือ วิธีนี้จะใช้ได้กับเฉพาะ ออบเจ็ค และไม่ใช้ session states

Random โหลดบาลานซิ่ง

วิธีนี้จะเลือกโฮสโดยวิธีการสุ่มจะใช้ได้เฉพาะกับออบเจ็ค โดยมีแนวโน้มที่จะกระจายข้อมูลที่เข้ามาอย่างเท่ากันระหว่างโฮส วิธีนี้ควรใช้ใน คลัสเตอร์ ที่แต่ละโฮสมีบริการต่างๆเหมือนกันและมีประสิทธิภาพเหมือนกัน ข้อเสียของวิธีนี้คือการเสียเวลาในการสร้างตัวเลขในการสุ่มขึ้นมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โหลดบาลานซิ่งสำหรับ HTTP Session

โหลดบาลานซิ่ง สำหรับ Servlets และ JSP HTTP session states สามารถทำโดยใช้วิธีดังนี้

-Built-in load-balancing capabilities of WebLogic proxy plug-ins การใช้ proxy plug-ins จะติดตั้งบนเว็บเซิร์ฟเวอร์ยี่ห้อใดก็ได้ แต่จะใช้ได้เฉพาะกับวิธี round-robin เท่านั้นในการกระจายคำร้องขอที่เป็น servlets และ JSP

-Load-balancing hardware เป็นการใช้โหลดบาลานเซอร์ที่เป็นฮาร์ดแวร์เข้ามาช่วยโดยจะสามารถใช้วิธีการทำโหลดบาลานซิ่งได้หลากหลาย

การรองรับเฟลโอเวอร์

เว็บโลจิกเซิร์ฟเวอร์ สามรถตรวจสอบความเสียหายของเซิร์ฟเวอร์ ได้จากวิธีต่อไปนี้

-Socket connection แบบต่อตรงกับ เซิร์ฟเวอร์ (Peer-to-peer)

-ข้อความ heartbeat

เซิร์ฟเวอร์ จะดูแลการใช้ IP socket ระหว่าง peer เซิร์ฟเวอร์ ถ้า เซิร์ฟเวอร์ ต่อเข้ากันแบบ peer เริ่มส่งข้อมูลโดยใช้ socket และเมื่อมีการปิดลงของการติดต่อ socket ก็จะรู้ว่า เซิร์ฟเวอร์ นั้นเสีย และงานบริการต่างที่เซิร์ฟเวอร์ นั้นทำจะถูกลบออกจาก JNDI naming tree การตรวจสอบยังสามารถใช้ heartbeat ในการตรวจสอบรายละเอียดของ heartbeat และ การติดต่อแบบsocket จะธิบายในหัวข้อต่อไป

การสื่อสารในคลัสเตอร์

เว็บโลจิกเซิร์ฟเวอร์ จะติดต่อกันระหว่างโฮสโดยใช้วิธีต่อไปนี้

-IP multicast

-IP socket

การสื่อสารแบบ IP Multicast-One-to-Many

เว็บโลจิกเซิร์ฟเวอร์ ใช้ IP multicast ในการตรวจสอบความล้มเหลวของ เซิร์ฟเวอร์ และดูแล Cluster-wide JNDI tree IP multicast เป็นการ broadcast โดยใช้ IP address ตั้งแต่ 239.255.255.255 ถึง 224.0.0.0 IP multicast จะไม่รับประกันความสูญหายของข้อความ

เว็บโลจิกเซิร์ฟเวอร์ ใช้ IP multicast ในการทำงานต่อไปนี้

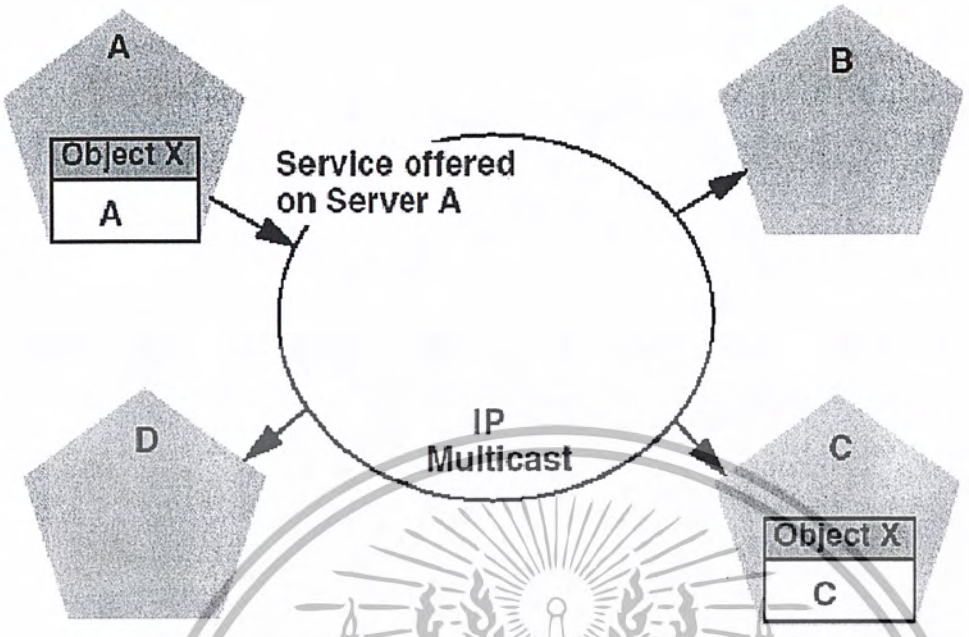
-Cluster-wide JNDI updates

-Cluster heartbeats

Cluster-Wide JNDI Updates

ทุกเซิร์ฟเวอร์ ใช้ IP multicast ในการประกาศว่าสามารถใช้ออบเจ็คได้ มีการติดตั้งเพิ่มหรือมีการเอาตัวไหนออกบ้างเซิร์ฟเวอร์ จะตรวจสอบการประกาศแล้วปรับปรุง JNDI tree ใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



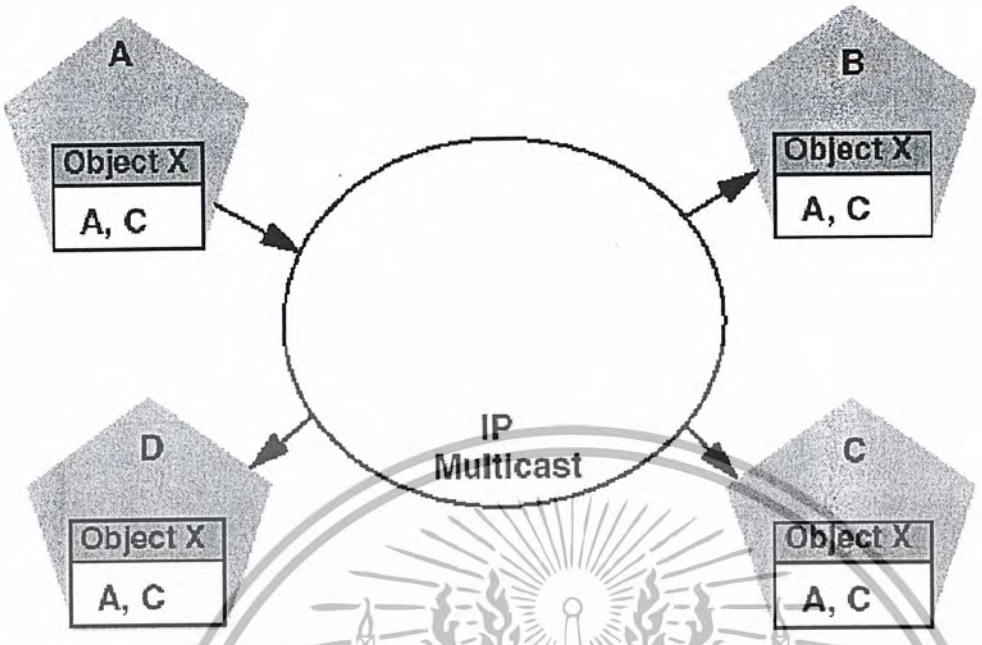
รูปที่ 3-1 การอัพเดท JNDI tree

การอัพเดท JNDI จะเกิดขึ้นได้ใน 2 กรณี

1. ถ้าการทำงานคลัสเตอร์ยังไม่ต่อเข้ากับ JNDI tree เซิร์ฟเวอร์จะต่อ replica-aware stub กับโลคอล (local) ออบเจ็คของเซิร์ฟเวอร์ก่อน เช่น เซิร์ฟเวอร์ A จะอัพเดทออบเจ็ค X โดย เซิรฟเวอร์ที่เหลือก็จะทำเหมือนกัน โดยเซิร์ฟเวอร์ A และ เซิร์ฟเวอร์ C จะมีออบเจ็ค X เหมือนกัน

2. ในกรณีที่คลัสเตอร์ทำงานแล้วจะมีการประกาศออบจ็คที่ตนเองมีให้เซิร์ฟเวอร์อื่นทราบแล้วทำการอัพเดทข้อมูลใหม่โดยเมื่อรับทราบครบทุกเซิร์ฟเวอร์แล้วจะได้ดังรูปที่ 3-2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-2 การอัพเดท JNDI tree ที่สมบูรณ์

Cluster Heartbeats

เว็บโลจิกเซิร์ฟเวอร์ ใช้การประกาศข้อความ Heartbeat ที่บอกถึงความคงอยู่ของ เซิร์ฟเวอร์ นั้น ในคลัสเตอร์ทุกเซิร์ฟเวอร์ รับข้อความ heartbeat เพื่อนำมาช่วยวิเคราะห์ว่าเซิร์ฟเวอร์เสียเมื่อไหร่

IP multicast สามารถเชื่อถือได้เมื่อส่งบนซับเน็ทเดียวกัน อย่างไรก็ตามถ้าต้องการแยกเซิร์ฟเวอร์ไว้ในที่อื่น จะมีโอกาสที่จะต้องผ่านหลายๆซับน์ท เพื่อรับประกันว่า IP multicast ถูกส่งอย่างถูกต้อง เน็ทเวิร์คต้องเป็นตามความต้องการดังนี้

1. ต้องสนับสนุน IP multicast packet propagation ทุกเราท์เตอร์ต้องตั้งค่าในการเผยแพร่ข้อความ multicast ไปยัง เซิร์ฟเวอร์ นั้น
2. ความล่าช้าของการส่งต้องน้อยเพื่อรับประกันว่าสามารถส่งข้อความ multicast ถึงได้ใน 200 ถึง 300 มิลลิวินาที
3. ค่า TTL (time to live) ต้องสูงเพียงพอที่จะรับประกันว่าเราท์เตอร์จะไม่ทิ้งแพคเก็ต ก่อนทีจะถึงปลายทาง สามารถตั้งค่าได้ใน *config.xml* ที่ค่า MulticastTTL ใน คลัสเตอร์ element

```
<คลัสเตอร์
    ...
    MulticastTTL="2"
/>
```

หรือใช้เว็บโลจิกคอนโซวในการตั้งค่า ค่า MulticastTTL เป็น 2 บอกว่าสามารถส่งผ่าน 2 เราท์เตอร์ได้ก่อนที่จะถูกทิ้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IP Socket-Peer-to-Peer Communication

เว็บโลจิกเซิร์ฟเวอร์ ใช้ IP socket กับการทำงานต่อไปนี้

-เพื่อเข้าถึงออบเจ็คที่ไม่ได้ทำคลัสเตอร์ จากระยะไกล

-เพื่อคัดลอก HTTP session states และ stateful session EJB states จากเซิร์ฟเวอร์หลักไปยังเซิร์ฟเวอร์ที่สอง เพื่อความสามารถในการใช้งานสูงขึ้นและรองเฟลโอเวอร์

-เพื่อเข้าถึงออบเจ็คที่ทำคลัสเตอร์จากระยะไกล

ปัจจัยสองอย่างที่กระทบประสิทธิภาพของการติดต่อแบบ socket คือ

-ใช้ native หรือ pure Java socket reader implementationหรือไม่

-ตั้งค่า socket reader threads ใน pure Java socket reader หรือไม่

Native socket reader implementation ให้ประสิทธิภาพดีที่สุดสำหรับการใช้งาน socket มากๆ Java socket reader รวมทุก socket แม้จะมีบางอันที่ไม่มีข้อมูล ส่วน native หรือ pure ตัว socket reader จะมีประสิทธิภาพกว่าเพราะจะรวบรวมเฉพาะ socket ที่ใช้ได้จริงและจะถูกบอกทันทีที่ socket ใช้ได้

<u>การติดต่อระหว่างผู้ใช้กับคลัสเตอร์</u>

ผู้ใช้เข้าถึงออบเจ็คหรือเซอร์วิสของเว็บโลจิกเซิร์ฟเวอร์ โดยใช้ JNDI-compliant naming เซอร์วิส ซึ่งเก็บรายการเซอร์วิสต่างที่ให้บริการ จัดการในรูปแบบ tree ตัวเซิร์ฟเวอร์เองก็ใช้ JNDI tree เพื่อเก็บเซอร์วิสบนเซิร์ฟเวอร์ เพื่อใช้ในการแสดงให้เซิร์ฟเวอร์อื่นในคลัสเตอร์เห็นว่ามีเซอร์วิส อะไรบ้างและมีเซอร์วิสอะไรที่ เซิร์ฟเวอร์อื่นมีบ้าง

แต่ละเซิร์ฟเวอร์สร้างและดูแลข้องมูลของ local JNDI tree ในตอนเริ่มแรกเซิร์ฟเวอร์จะใช้ local JNDI tree ก่อน แล้วส่งข้อมูลออบเจ็คที่มีไปให้ทุกเซิร์ฟเวอร์ในคลัสเตอร์โดยใช้ multicast address แล้ว เซิร์ฟเวอร์อื่นก็จะทำการ update JNDI tree

<u>การเกิดความขัดแย้งใน JNDI Naming</u>

การเกิดความขัดแย้งใน JNDI Naming จะเกิดเมื่อเซิร์ฟเวอร์พยายามใช้ชื่อ JNDI ของเซิร์ฟเวอร์ที่ไม่ใช่คลัสเตอร์ที่มีชื่อซ้ำกับที่มีใน JNDI ของเซิร์ฟเวอร์นั้น ความขัดแย้งใน Cluster-level JNDIเกิดเมื่อเซิร์ฟเวอร์พยายามใช้ชื่อออบเจ็คของเซิร์ฟเวอร์ที่ไม่คลัสเตอร์ ที่มีอยู่แล้วใน JNDI tree การจัดการความสับสนต่างๆทำได้ดังนี้

ถ้าสองเซิร์ฟเวอร์ในคลัสเตอร์พยายามใช้ ออบเจ็คที่ต่างกันโดยใช้ชื่อเดียวกัน ทั้งสองเซิร์ฟเวอร์ จะสามารถทำได้เฉพาะสองเซิร์ฟเวอร์นั้น อย่างไรก็ตามแต่ละเซิร์ฟเวอร์นั้นจะปฏิเสธการเก็บชื่อซ้ำลงบน JNDI tree การสับสนแบบนี้จะคงอยู่จนกว่าหนึ่งในสองเซิร์ฟเวอร์นั้นปิดลงหรือหนึ่งในนั้นปิด ออบเจ็ค ตัวนั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

J2EE คลัสเตอร์ริ่ง

J2EE สามารถให้บริการ คลัสเตอร์ริ่ง ได้หลายทาง และ เว็บโลจิกเซิร์ฟเวอร์ ก็สามารถแบ่งการทำงานของ คลัสเตอร์ริ่ง ได้ดังนี้

- JSP/sevlets คลัสเตอร์ริ่ง
- ออบเจ็ค คลัสเตอร์ริ่ง
- Java Message เซอร์วิส (JMS) เซิร์ฟเวอร์ คลัสเตอร์ริ่ง

HTTP JSP/Servlets คลัสเตอร์ริ่ง

เว็บโลจิกเซิร์ฟเวอร์สามารถรองรับคลัสเตอร์ริ่งบนJSP และ servlet โดยการทำสำเนาข้อมูลของผู้ใช้ที่ทำการติดต่อกับเซิร์ฟเวอร์ผ่านทาง JSP และ servlets และเพื่อแน่ใจว่าการทำงานของเซสชั่นมีจริง สถานะของเซสชั่นต้องถูกป้องกัน และสถานะนี้สามารถถูกป้องกันได้หลายทางคือ

- In-memory replication
- File-based persistence
- JDBC-based persistence

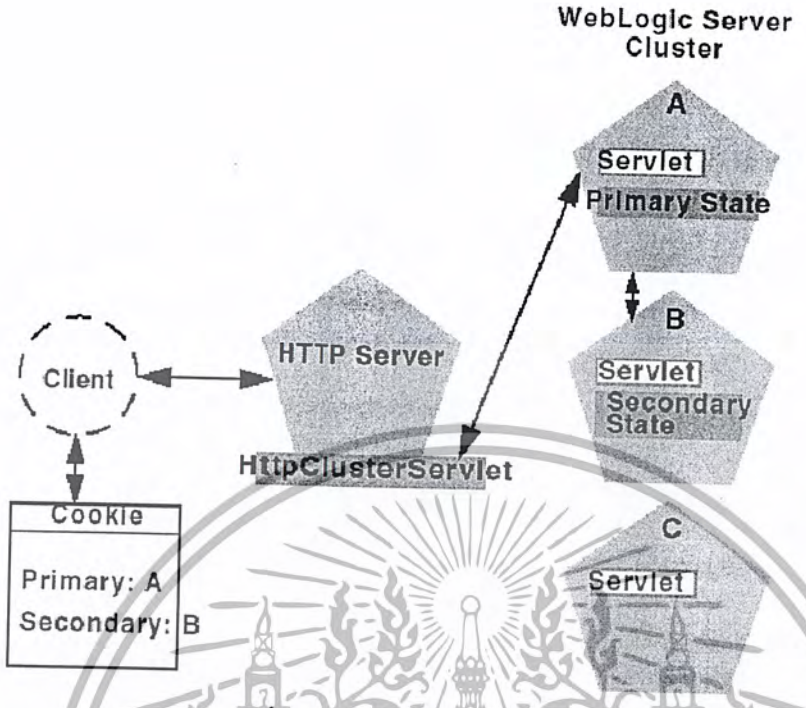
In-Memory Replication

นี่เป็นการทำงานแบบทำสำเนาของสถานะของเซสชั่นจากเซิร์ฟเวอร์นึงไปยังอีกเซริ์ฟเวอร์นึง เมื่อไคลเอนท์ติดต่อมายัง เว็บโลจิค เซิร์ฟเวอร์ และทำงานเกี่ยวกับ JSP / servlets เซสชั่นถูกสร้างขึ้นบนเซิร์ฟเวอร์ที่ไคลเอนท์ติดต่อเข้ามาดังรูปที่ 3-3 สถานะนี้เรียกว่า primary session state ส่วนสถานะที่ถูกสำเนาไปยังเครื่องต่างๆ ในคลัสเตอร์เราเรียกว่า secondary session state ดังนั้น ถ้าเซิร์ฟเวอร์ที่ต้องตอบกลับไคลเอนท์นั้นใช้งานไม่ได้ ก็จะมีการป้องกันการเฟลโอเวอร์โดยการให้เซิร์ฟเวอร์ตัวอื่นที่ได้ secondary session state ทำการ เฟลโอเวอร์โดยอัตโนมัติ

และเพื่อเพิ่มประสิทธิภาพในการใช้ in-memory replication กับ HTTP session state เว็บโลจิกเซิร์ฟเวอร์ ต้องรองรับการทำงานดังนี้อีกด้วยคือ

- ผ่านทางโหลดบาลานซิ่งฮาร์ดแวร์
- ผ่านทางกลุ่มของเว็บเซิร์ฟเวอร์โดยใช้ เว็บโลจิกพรอกซี่เข้ามาช่วยในการติดต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3-3 การสำรองเซสชั่นของผู้ใช้

ความต้องการของโหลดบาลานเซอร์

โหลดบาลานซิ่งฮาร์ดแวร์ต้องรองรับการทำงานของ in-memory replication ดังนี้คือ

- การเข้ารหัส SSL
- สามารถรองรับคุ้กกี้ได้

เว็บโลจิก เซิร์ฟเวอร์ ต้องรองรับการเขียนคุ้กกี้ผ่านโหลดบาลานเซอร์ไปที่เครื่องไคลเอนท์ และ บาลานเซอร์ทำการควบคุมเครื่องไคลเอนท์ผ่านทางการเขียนคุ้กกี้ที่เครื่องไคลเอนท์โดยใช้ HTTP session state

และเมื่อมี SSL เข้ามา เครื่องที่ทำหน้าที่เป็นโหลดบาลานเซอร์จะทำการเข้ารหัสและถอดรหัสระหว่างเครื่องไคลเอนท์และเว็บโลจิกเซิร์ฟเวอร์และโหลดบาลานเซอร์จะใช้ plain text ที่เว็บโลจิก เซิร์ฟเวอร์ ใช้กับเครื่องไคลเอนท์ มาเพื่อควบคุมการติดต่อกันระหว่างเครื่องไคลเอนท์และเครื่องเซิร์ฟเวอร์อื่นๆที่อยู่ในคลัสเตอร์

Proxy Requirements

เว็บโลจิกพรอกซี่ ทำการติดต่อเพื่อควบคุมการทำงานของ เว็บโลจิก เซิร์ฟเวอร์ แทน ซึ่งเป็นโฮสต์ที่ใช้งานเกี่ยวกับ Servlets และ JSP และทำการส่งข้อมูล HTTP request ไปที่เครื่องเซิร์ฟเวอร์โดยใช้การทำงานแบบ round-robin และพรอกซี่ ยังทำหน้าที่ในการแบ่งการทำงานที่เครื่องเซิร์ฟเวอร์ส่งไปให้เครื่องไคลเอนท์ได้อีกด้วยถ้า เว็บโลจิก เซิร์ฟเวอร์ เกิดการทำงานผิดพลาด และเว็บโลจิก เซิร์ฟเวอร์ สามารถรองรับการทำงานของเว็บเซิร์ฟเวอร์ และพรอกซี่ซอร์ฟแวร์ดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

- Netscape Enterprise เซิร์ฟเวอร์โดยมี Netscape (proxy) plug-ins
- Apache เซิร์ฟเวอร์โดยมี (proxy) plug-ins
- Microsoft Internet Information เซิร์ฟเวอร์โดยมี Microsoft-IIS (proxy) plug-ins
- เว็บโลจิกเซิร์ฟเวอร์โดยมี HttpClusterServlet

Configuring IN-Memory HTTP Replication

ทุกทุกเว็บแอพพลิเคชันที่ทำงานบน เว็บโลจิก เซิร์ฟเวอร์ จะมี เว็บโลจิก Deployment Descriptor(weblogic.xml) ผู้ใช้ควรที่จะเซตค่าพารามิเตอร์ *PersistentStoreType* ให้ทำการเก็บค่าใน *weblogic.xml* เพื่อเป็นการเปิดใช้งานของ in-memory HTTP session replication

File-Based Replication

File-Based Replication ทำหน้าที่ในการเก็บค่าเซสชั่นต่างๆในไฟล์

Configuring File-Based Session Persistence

Configuring File-Based Session Persistence สามารถถูกคอนฟิกโดยการใช้ค่าแอตทริบิวต์ 2 ค่าใน เว็บโลจิก Deployment Descriptor(weblogic.xml) ซึ่งทั้ง 2 ค่านั้นคือ

- ข้อมูลของ PersistentStoreType ที่อยู่ในส่วนของ <session-descriptor> ซึ่งอยู่ภายในไฟล์
- ข้อมูลของ PersistenStoreDir ซึ่งเป็นส่วนที่ควรเก็บไว้ในเซิร์ฟเวอร์

JDBC-Based Session Persistence

JDBC-Based Session Persistence จะเก็บข้อมูลเซสชั่นต่างๆลงในฐานข้อมูล

กาตั้งค่า JDBC-Based Session Persistence

คือการปรับค่าต่างๆที่อยู่ในไฟล์ weblogic.xml

- เซตคุณลักษณะ PersistentStoreType ที่อยู่ในส่วน <session-descriptor> และ JDBC ที่อยู่ใน เว็บโลจิก Deployment Descriptor(weblogic.xml)
- เซตการเชื่อมต่อที่ต้องการใช้ (เซตค่าการเชื่อมต่อ JDBC ที่ถูกใช้เพื่อเก็บข้อมูลกับ PersistentStorePool)
- เซตค่า Access Control List(ACL) เพื่อการเชื่อมต่อและเช็คการมีสิทธิการใช้งานของผู้ใช้ต่างๆ
- เซตค่าตารางในฐานข้อมูลที่มีชื่อว่า w1_servlet_session สำหรับ JDBC-based persistence การเชื่อมต่อนี้จะต่อกับฐานข้อมูลที่ซึ่งต้องการใช้งานในการเขียนหรืออ่านตารางนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
55089

ออบเจ็คคลัสเตอร์ริ่ง

ออบเจ็คคลัสเตอร์ เป็นออบเจ็คที่ใช้งานโดยที่เครื่องไคลเอนท์ใช้ Remote Method Invocation(RMI) ยกตัวอย่างเช่น EJB ออบเจ็คคลัสเตอร์สามารถทำงานบน เว็บโลจิกคลัสเตอร์ได้และสามารถทำงานบนเว็บโลจิกเซิร์ฟเวอร์ได้อีกด้วย

เมื่อ EJB ที่รองรับการทำคลัเตอร์ถูกคอมไพล์ *ejbc*(เป็นคอมไพเลอร์จาก BEA) จะทำการส่งค่าผ่านทาง อินเทอร์เฟสของ EJB ผ่านทาง *rmic*(RMI ออบเจ็ค compiler tool) และคอมไพล์เพื่อที่จะทำเป็นค่าสตับเพื่อนำมาใช้ใน EJB replica-aware stub เป็นสตับที่ใช้งานกับเครื่องไคลเอนท์ของ EJB แต่ replica-aware stub บรรจุการทำงานของ EJB หรือ RMI ที่อยู่บน เว็บโลจิก เซิร์ฟเวอร์ เมื่อเครื่องไคลเอนท์ทำการติดต่อกับคลัสเตอร์ออบเจ็ค แล้ว replica-aware stub จะถูกส่งไปที่เครื่องไคลเอนท์ โดยสตับอันนี้จะบรรจุฟังก์ชันการทำงานของโหลดบาลานซิ่งเพื่อใช้ในการเรียกออบเจ็ค และในแต่ละครั้งสตับจะใช้ฟังก์ชันโหลดบาลานซิ่งของตนเองเพื่อเลือกว่าตัวไหนจะถูกเรียก แล้วก็จะทำการส่งไปยังเครื่องไคลเอนท์ และในกรณีที่ไม่สามารถทำงานได้สตับจะตัดการทำงานแล้วพยายามติดต่อกับอีกเซิร์ฟเวอร์เพื่อทำการ เฟลโอเวอร์

คลัสเตอร์ EJBs

EJB สามารถที่จะมี replica-aware stub ได้ 2 อย่างเพื่อที่จะทำเป็น EJBHome และ EJBObject ดังนั้น EJB สามารถรู้ถึงผลที่เกิดของโหลดบาลานซิ่งและการเฟลโอเวอร์ ใน 2 ระดับคือ

- client look-up time
- client method invocation time

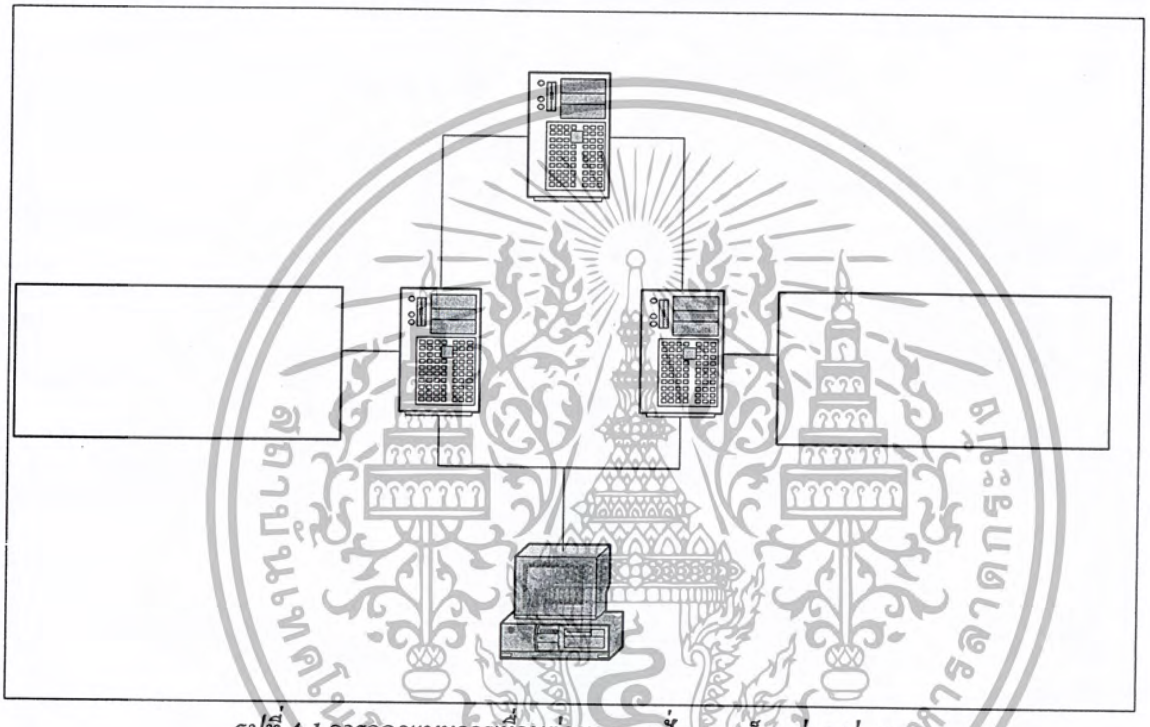
เมื่อ EJB ทำงานบนเซิร์ฟเวอร์ เซิร์ฟเวอร์นั้นจะถูกล้อมรอบไปด้วย Cluster-wide naming เซอร์วิส เพราะว่า EJB home สามารถทำการคลัสเตอร์ได้ และแต่ละเซิร์ฟเวอร์นำมารวมกันโดยใช้ชื่อเดียวกันได้เลย และเมื่อเครื่องไคลเอนท์มองหา มันจะได้ค่าที่เป็นreplica-aware stub ซึ่งอ้างอิงกับข้อมูลที่อยู่จริงได้ และเมื่อ home ต้องการที่จะหาหรือสร้าง bean มันจะต้องสร้างขึ้นมาบนเครื่อง local เซิร์ฟเวอร์ และส่งค่าสตับคืนมายังที่เซิร์ฟเวอร์เดิมและทำการ เฟลโอเวอร์ เฉพาะที่ระดับเดิมเท่านั้น เพราะว่ามันเป็นไปไม่ได้ที่จะทำทั้งหมด เนื่องจากค่า instance จะต้องอ่านค่าจากฐานข้อมูลก่อนการส่งจะเริ่มต้นขึ้นและก่อนการ commit

และเมื่อ home ค้นหาหรือสร้าง bean ได้แล้ว มันจะส่งค่า replica-aware stub ซึ่ง stub load balance ของทุกทุกการเรียกนั้นไม่สามารถทำการแก้ไขการเฟลโอเวอร์อย่างอัตโนมัติทุกตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การออกแบบและติดตั้งสถาปัตยกรรม .NET Architecture

4.1 ขั้นตอนการออกแบบ



รูปที่ 4-1 การออกแบบการเชื่อมต่อของทางฝั่งดอทเน็ทอย่างคร่าวๆ

จากรูป ในขณะที่เครื่อง Client ติดต่อเข้ามา จะเข้ามาทางเครื่องเซิร์ฟเวอร์ 1 และเครื่องเซิร์ฟเวอร์ 2 ที่ทำโหลดบาลานซ์กันอยู่ และเครื่องทั้งสองก็จะไปติดต่อกับดาต้าเบสเซิร์ฟเวอร์อีกที

โดยที่คุณลักษณะของเครื่องที่ใช้ทำการทดลองเป็นดังนี้

เซิร์ฟเวอร์ 1(Server 1)

- ใช้ ซีพียู เพนเทียม โฟว์ 2.4 กิ๊กกะเฮิร์ตซ์
- ใช้ แรม ขนาด 512 เมกกะไบต์
- ใช้ Windows 2000 Advanced Server เป็นระบบปฏิบัติการหลัก
- ชื่อเครื่อง OLALA111
- ไอพีของเครื่อง 161.246.6.111
- ไอพีกลาง 161.246.6.102

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เซิร์ฟเวอร์ 2(Server 1)

- ใช้ ซีพียู เพนเทียม โฟว์ 1.6 กิ๊กกะเฮิร์ตซ์
- ใช้ แรม ขนาด 256 เมกกะไบต์
- ใช้ Windows 2000 Advanced Server เป็นระบบปฏิบัติการหลัก
- ชื่อเครื่อง OLALA124
- ไอพีของเครื่อง 161.246.6.124
- ไอพีกลาง 161.246.6.102

ดาต้าเบสเซิร์ฟเวอร์(Database Server)

- ใช้ซีพียู ดูรอน 900 เมกกะเฮิร์ตซ์
- ใช้แรมขนาด 256 เมกกะไบต์
- ใช้ Windows 2000 Advanced Server เป็นระบบปฏิบัติการหลัก
- ใช้ Microsoft SQL Server 2000 เป็น ดาต้าเบสเซิร์ฟเวอร์
- ชื่อเครื่อง OLALA127
- ไอพีของเครื่อง 161.246.6.127

ข้อดีในการออกแบบ

1. สามารถรองรับปริมาณงานได้มากขึ้น
2. สามารถรองรับการ Fail over ของเซิร์ฟเวอร์
3. สามารถเพิ่มเติมเซิร์ฟเวอร์ได้โดยผู้ใช้งานไม่รู้
4. มีงานดูแลรักษาที่ง่ายโดยสามารถเปิดและปิดเซิร์ฟเวอร์โดยไม่กระทบต่อการทำงานรวม
5. มีการแบ่งงานอย่างทั่วถึงจึงใช้ประสิทธิภาพทั้งฮาร์ดแวร์และซอร์ฟแวร์ได้อย่างเต็มที่

ข้อเสียในการออกแบบ

1. ไม่สามารถเลือกวิธีการโหลดบาลานซิ่งได้เอง
2. ขาดความปลอดภัยเนื่องจากไม่มีไฟร์วอลและเป็น DMZ โซนที่ทุกคนเข้าถึงได้
3. ยึดติดกับโปรแกรมของไมโครซอฟท์ โดยใช้ได้เฉพาะกับ IIS เซิร์ฟเวอร์

ข้อควรปรับปรุง

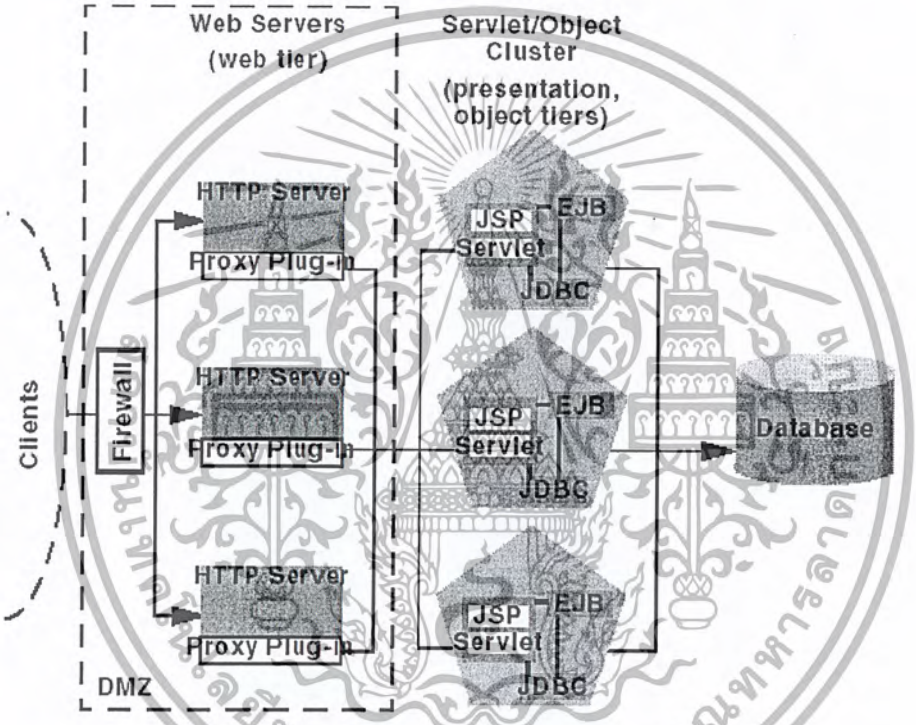
1. การเพิ่มจำนวนเครื่องจะช่วยเพิ่มประสิทธิภาพการทำงานได้
2. เพื่อความปลอดภัยสามารถเพิ่มไฟร์วอลได้โดยไม่กระทบการทำงานรวม
3. การต่อเซิร์ฟเวอร์กับสวิทช์จะมีประสิทธิภาพกว่าฮับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การออกแบบและติดตั้งสถาปัตยกรรม J2EE Architecture

5.1 การออกแบบโลจิคอลวิว (Logical View)

สถาปัตยกรรมที่เลือกใช้เป็นแบบ สถาปัตยกรรมแบบทูเทียร์(Two-Tier Architecture) โดยวิธีแบบสถาปัตยกรรมพรอกซี (Proxy Architectures) ที่มี HTTP พรอกซีเซิร์ฟเวอร์อยู่ในชั้นเว็บเทียร์ (Web tier) ทำหน้าที่ส่งข้อมูลไปให้ชั้นพรีเซ็นเตชั่นและออบเจ็ค (Presentation and Object tier) ทำงานดัง รูปที่ 5-1



รูปที่ 5-1 Two-Tier Proxy Architecture

สถาปัตยกรรมแบบทูเทียร์ ประกอบด้วย 2 เลเยอร์หลักของฮาร์ดแวร์และซอร์ฟแวร์ ได้แก่ เว็บเทียร์และออปเจ็คเทียร์

5.1.1เว็บเทียร์

สถาปัตยกรรมแบบพรอกซี่จะใช้ประโยชน์ของฮาร์ดแวร์และซอร์ฟแวร์ออย่างเต็มที่ ในชั้นนี้ สามารถมีเว็บเซิร์ฟเวอร์ได้มากกว่าหนึ่งโดยจะอาศัยไฟร์วอลหรือโหลดบาลานเซอร์(Load Balancer) ในการทำโหลดบาลานซิ่งเพื่อจ่ายงานให้ เว็บเซิร์ฟเวอร์ในชั้นนี้สามารถเป็นยี่ห้ออะไรก็ได้ไม่จำเป็นต้องเป็นเว็บโลจิก โดยใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทำงานในชั้นนี้จะดูแลเฉพาะเว็บเพจธรรมดาโดยจะไม่ยุ่งเกี่ยวกับ JSP, servlets และ EJB ซึ่งจะส่งให้ชั้นออบเจ็คจะเป็นคนจัดการ (สามารถใช้ชั้นนี้ทำการจ่ายงานแบบโหลดบาลานซิ่ง)

5.1.2 ชั้นพรีเซ็นเตชั่นและออบเจ็คเทียร์

ในชั้นนี้จะใช้การคลัสเตอร์ของเว็บโลจิกเซิร์ฟเวอร์หลายๆตัวโดนเซิร์ฟเวอร์นี้สามารถติดตั้งลงบนเครื่องเดียวกันหรือคนละเครื่องก็ได้ โดยจะดูแลออบเจ็คต่างๆและมีการแบ่งงานแบบโหลดบาลานซิ่ง

5.1.3 ข้อดีของสถาปัตยกรรมพรอกซี

1. ใช้งานฮาร์ดแวร์ที่มีอยู่ให้เกิดประโยชน์สูงสุดโดยไม่ต้องซื้อโหลดบาลานเซอร์ และในกรณีที่มีการติดตั้งเว็บไว้อยู่แล้ว (ที่ไม่ใช่แบบไดนามิก) ก็ไม่จำเป็นต้องเปลี่ยเซิร์ฟเวอร์โดยสามารถใช้งานเข้ากับเว็บโลจิกเซิร์ฟเวอร์ได้
2. ไม่ต้องตั้งกฎให้แก่ไฟร์วอลใหม่และสามารถกันการเข้าถึงเว็บโลจิกเซิร์ฟเวอร์โดยตรงจากภายนอกได้

5.1.4 ข้อจำกัดของสถาปัตยกรรมพรอกซี

1. การดูแลลำบากมากขึ้นเพราะไม่สามารถดูแลผ่านเว็บโลจิกเซิร์ฟเวอร์ได้โดยตรง และการตั้งค่าหรือติดตั้งลำบากมากขึ้น
2. รูปแบบการทำโหลดบาลานซิ่งจำกัดเพราะการใช้ Proxy Plug-ins

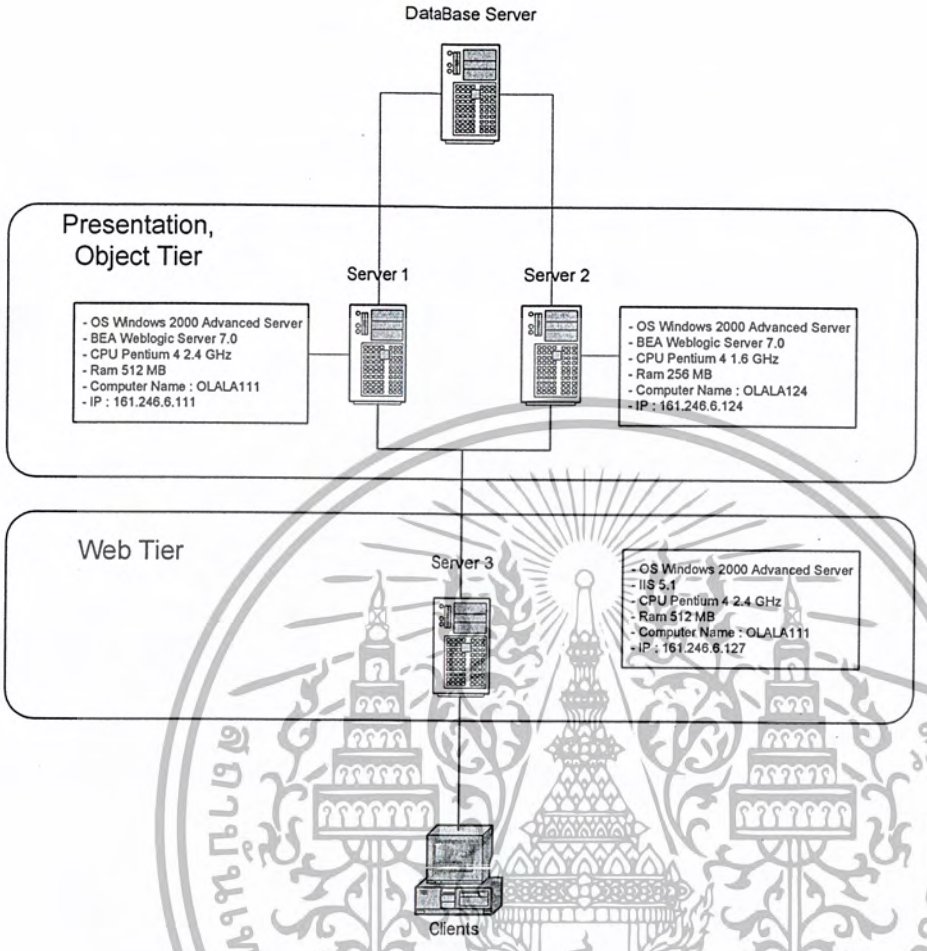
5.1.5 เปรียบเทียบ Proxy Plug-ins กับโหลดบาลานเซอร์

การใช้โหลดบาลานเซอร์กับเว็บโลจิกโดยตรงมีข้อดีหลายอย่าง อย่างแรก โหลดบาลานเซอร์ไม่จำเป็นต้องติดตั้งและดูแลชั้นของ HTTP และไม่จำเป็นต้องติดตั้ง Proxy Plug-ins การที่ตัดชั้นที่เป็น พรอกซีออกทำให้ลดปริมาณการเชื่อมต่อลง

โหลดบาลานเซอร์ยังมีความยืดหยุ่นในการเลือกวิธีทำโหลดบาลานซิ่งได้หลากหลายกว่า โดยถ้าใช้ Proxy Plug-ins จะจำกัดอยู่ที่วิธี round-robin แต่โหลดบาลานเซอร์ต้องเป็นแบบที่แน่ใจได้ว่าสามารถรักษาสถานการณ์เชื่อมต่อได้เช่นการใช้เซ็ดชั่น (Session)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2 การออกแบบฟิสิกคอลวิว (Physical View)



รูปที่ 5-2 สถาปัตยกรรม J2EE ที่ออกแบบ

เซิร์ฟเวอร์ 1 (Server 1)

- ใช้ ซีพียู เพนเทียม โฟว์ 2.4 กิ๊กกะเฮิร์ตซ์
- ใช้ BEA WebLogic Server 7.0 เป็นเว็บเซิร์ฟเวอร์
- ใช้ แรม ขนาด 512 เมกกะไบต์
- ใช้ Windows 2000 Advanced Server เป็นระบบปฏิบัติการหลัก
- ชื่อเครื่อง OLALA111
- ไอพีของเครื่อง 161.246.6.127

เซิร์ฟเวอร์ 2 (Server 2)

- ใช้ ซีพียู เพนเทียม โฟว์ 1.6 กิ๊กกะเฮิร์ตซ์
- ใช้ BEA WebLogic Server 7.0 เป็นเว็บเซิร์ฟเวอร์
- ใช้ แรม ขนาด 256 เมกกะไบต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ใช้ Windows 2000 Advanced Server เป็นระบบปฏิบัติการหลัก
- ชื่อเครื่อง OLALA124
- ไอพีของเครื่อง 161.246.6.124

เซิร์ฟเวอร์ 3 (Server 3)

- ใช้ ซีพียู เพนเทียม โฟว์ 2.4 กิ๊กกะเฮิร์ตซ์
- ใช้ IIS 5.1 เป็นเว็บเซิร์ฟเวอร์
- ใช้ แรม ขนาด 512 เมกกะไบต์
- ใช้ Windows 2000 Advanced Server เป็นระบบปฏิบัติการหลัก
- ชื่อเครื่อง OLALA111
- ไอพีของเครื่อง 161.246.6.111

ดาต้าเบสเซิร์ฟเวอร์(Database Server)

- ใช้ Mysql เป็น ดาต้าเบสเซิร์ฟเวอร์

5.2.1 ข้อดีในการออกแบบ

1. สามารถรองรับปริมาณงานได้มากขึ้น
2. สามารถรองรับการ Fail over ในชั้นชั้นพรีเซ็นเตชั่นและออบเจ็คเทียร์
3. การทำโหลดบาลานที่เซิร์ฟเวอร์ 1 และ 2 มีประสิทธิภาพมากและสามารถเลือกวิธีได้หลากหลาย
4. มีการแบ่งงานอย่างทั่วถึง
5. ใช้ประโยชน์จากฮาร์ดแวร์และซอร์ฟแวร์ได้อย่างเต็มประสิทธิภาพ
6. สามารถทำงานเว็บโลจิกเซิร์ฟเวอร์ได้หลายตัวในเครื่องเดียว
7. สามารถปรับปรุงเพิ่มเติมระบบในอนาคตได้ง่าย

5.2.2 ข้อเสียในการออกแบบ

1. เนื่องจากในชั้นเว็บเทียร์ไม่สามารถใช้เครื่องคอมพิวเตอร์หลายเครื่องในการรองรับการ Fail over ได้เนื่องจากใช้เครื่องเซิร์ฟเวอร์ 3 เพียงตัวเดียวเป็นโหลดบาลานเซอร์
2. การทำงานต้องผ่านเซิร์ฟเวอร์ 3 ก่อน จึงจะทำงานจริงที่ เซิร์ฟเวอร์1 และ 2 จึงทำให้ใช้ปริมาณการเชื่อมต่อมาขึ้น
3. การทำโหลดบาลานที่เซิร์ฟเวอร์ 3 จำกัดที่วิธี Round-Robin วิธีเดียว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2.3 ข้อควรปรับปรุง

1. การใช้โหลดบาลานเซอร์แบบฮาร์ดแวร์แทนที่เซิร์ฟเวอร์ 3 จะทำให้การทำงานมีประสิทธิภาพมากขึ้น
2. เพิ่มเซิร์ฟเวอร์ในชั้นเว็บเทียร์ให้มากขึ้นได้โดยมีโหลดบาลานเซอร์คอยจ่ายงานให้เพื่อป้องกันการ Fail over ที่ชั้นเว็บเทียร์
3. ควรมีการใช้ไฟร์วอลเพื่อเพิ่มความปลอดภัย
4. สามารถเพิ่มความปลอดภัยของเว็บโลจิกเซิร์ฟเวอร์ได้โดยให้ เซิร์ฟเวอร์ในเว็บเทียร์เท่านั้นที่จะติดต่อกับเว็บโลจิกเซิร์ฟเวอร์ได้โดยอาจใช้ Private IP ก็ได้
5. สามารถให้ DNS เซิร์ฟเวอร์ช่วยในการทำโหลดบาลานซิ่งโดยใช้โดเมนเนมเข้ามาช่วยได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
การทดสอบประสิทธิภาพ

สภาพแวดล้อมในการทดสอบ

เครื่องเซิร์ฟเวอร์ที่ใช้

เครื่องที่ 1 (161.246.6.111)

- CPU Pentium4 2.4 GHz
- RAM 512 MB
- OS windows 2000 Advanced Server

เครื่องที่ 2 (161.246.6.124)

- CPU Pentium4 1.6 GHz
- RAM 256 MB
- OS windows 2000 Advanced Server

โปรแกรมที่ใช้วัดประสิทธิภาพ

Microsoft Application Center Test

ค่าที่วัดประสิทธิภาพ

- Total Request (Hit)
- Output Request Per Second (RPS)
- Average Bandwidth (Bytes/sec)
- Average Rate of Sent Bytes (Bytes/sec)
- Average Rate of Receive Bytes (Bytes/sec)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เราแบ่งการทดสอบออกเป็น 6 การทดสอบดังนี้คือ

การทดสอบ Load Balancing บน IIS Server 5.1

การทดสอบที่ 1 การใช้งาน static web page (html)

การทดสอบที่ 1.1

ส่ง Http Get ไปยังเซิร์ฟเวอร์ ที่ทำ load balancing กัน แล้วเก็บผลการทดสอบ

การทดสอบที่ 1.2

ส่ง Http Get ไปยังเซิร์ฟเวอร์โดยทดสอบทีละเซิร์ฟเวอร์ โดยใช้ Load Level=100 แล้วเก็บผลการทดสอบ

การทดสอบที่ 2 การใช้งาน dynamic web page (ASP)

การทดสอบที่ 2.1

ส่ง Http Get ไปยังเซิร์ฟเวอร์ ที่ทำ load balancing กัน แล้วเก็บผลการทดสอบ

การทดสอบที่ 2.2

ส่ง Http Get ไปยังเซิร์ฟเวอร์ โดยทดสอบทีละเซิร์ฟเวอร์ โดยใช้ Load Level =100 แล้วเก็บผลการทดสอบ

การทดสอบที่ 3 การใช้งานกับสภาพการใช้งานจริงโดยมี static และ dynamic web page ร่วมถึงการเชื่อมต่อ DBMS Server โดยแบ่งเป็น static web = 80% และ dynamic web = 20%

การทดสอบที่ 3.1

ส่ง Http Get ไปยังเซิร์ฟเวอร์ ที่ทำ load balancing กัน แล้วเก็บผลการทดสอบ

การทดสอบที่ 3.2

ส่ง Http Get ไปยังเซิร์ฟเวอร์ โดยทดสอบทีละเซิร์ฟเวอร์ โดยใช้ Load Level =100 แล้วเก็บผลการทดสอบ

การทดสอบ Load Balancing บน Bea WebLogic Server 7.0

การทดสอบที่ 4 การใช้งาน static web page (html)

การทดสอบที่ 4.1

ส่ง Http Get ไปยังเซิร์ฟเวอร์ที่ทำ load balancing และ cluster กัน แล้วเก็บผลการทดสอบ

การทดสอบที่ 4.2

ส่ง Http Get ไปยังเซิร์ฟเวอร์ โดยทดสอบทีละเซิร์ฟเวอร์ โดยใช้ Load Level =100 แล้วเก็บผลการทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดสอบที่ 5 การใช้งาน dynamic web page (jsp, ejb, servlet)

การทดสอบที่ 5.1

ส่ง Http Get ไปยังเซิร์ฟเวอร์ ที่ทำ load balancing และ cluster กันแบ่งเป็นแบบ Stateless และ Stateful แล้วเก็บผลการทดสอบ

การทดสอบที่ 5.2

ส่ง Http Get ไปยังเซิร์ฟเวอร์ โดยทดสอบทีละเซิร์ฟเวอร์ แบ่งเป็นแบบ Stateless และ Stateful โดยใช้ Load Level =100 แล้วเก็บผลการทดสอบ

การทดสอบที่ 6 การใช้งานกับสภาพการใช้งานจริงโดยมี static และ dynamic web page ร่วมถึงการเชื่อมต่อ DBMS Server โดยแบ่งเป็น static web=80% และ dynamic web=20%

การทดสอบที่ 6.1

ส่ง Http Get ไปยังเซิร์ฟเวอร์ ที่ทำ load balancing และ cluster กันแล้วเก็บผลการทดสอบ

การทดสอบที่ 6.2

ส่ง Http Get ไปยังเซิร์ฟเวอร์โดยทดสอบทีละเซิร์ฟเวอร์โดยใช้ Load Level =100 แล้วเก็บผลการทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลการทดสอบ

การทดสอบที่	Load Level	Total Request (Hit)	Output (RPS)	Average Bandwidth (bytes/sec)	Average rate of sent bytes (bytes/sec)	Average rate of receive bytes (bytes/sec)
การทดลองที่ 1.1	1	8,841	73.68	1,299,131	28,494	1,270,637
	100	10,763	89.69	1,581,547	34,703	1,536,844
	500	11,176	93.13	1,642,151	35,799	1,606,351
	1000	13,833	115.28	2,032,311	43,925	1,988,385
การทดลองที่ 1.2	1	4,354	36.28	639,840	14,022	625,817
	100	6,265	52.21	920,661	20,217	900,444
	500	6,028	50.23	885,863	19,317	966,545
	1000	5,614	46.78	825,064	17,801	807,262
การทดลองที่ 2.1	1	5,684	47.37	835,195	18,245	815,949
	100	8,672	72.27	1,274,269	27,919	1,246,350
	500	11,532	96.10	1,694,427	36,941	1,657,485
	1000	14,274	118.95	2,097,091	45,335	2,051,756

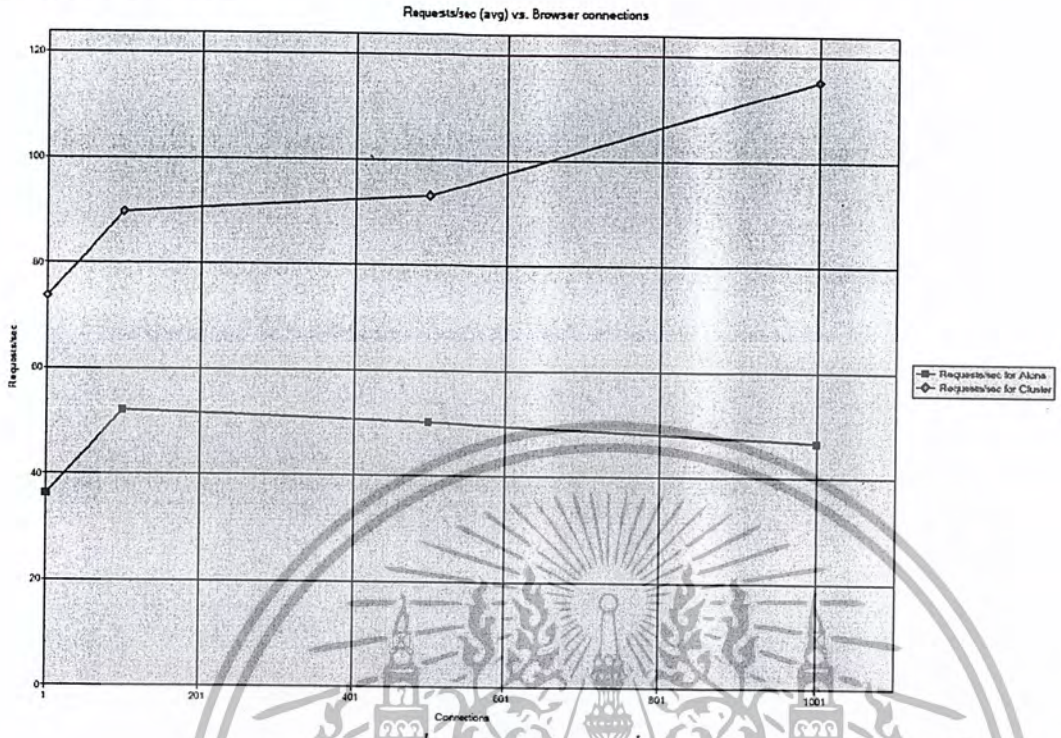
การทดสอบที่	Load Level	Total Request (Hit)	Output (RPS)	Average Bandwidth (bytes/sec)	Average rate of sent bytes (bytes/sec)	Average rate of receive bytes (bytes/sec)
การทดลองที่ 2.2	1	5,488	45.73	806,481	17,698	788,783
	100	5,874	48.95	863,204	18,949	844,254
	500	3,384	28.20	497,315	10,803	486,511
	1000	4,944	41.20	726,602	15,653	710,949
การทดลองที่ 3.1	1	832	6.93	12,460	2,531	9,928
	100	845	7.04	12,746	2,598	10,147
	500	987	8.23	15,271	2,897	12,374
	1000	1,259	10.49	19,658	3,609	16,049
การทดลองที่ 3.2	1	896	7.47	11,981	2,842	9,138
	100	545	4.54	7,295	1,693	5,602
	500	563	4.69	7,566	1,614	5,951
	1000	459	3.83	6,173	1,292	4,880

การทดสอบที่	Load Level	Total Request (Hit)	Output (RPS)	Average Bandwidth (bytes/sec)	Average rate of sent bytes (bytes/sec)	Average rate of receive bytes (bytes/sec)
การทดลองที่ 4.1	1	52,495	437.46	218,729	146,548	72,180
	100	76,632	638.60	319,300	213,931	105,369
	500	58,748	489.57	244,783	164,004	80,778
	1000	36,928	307.73	153,866	103,090	50,776
การทดลองที่ 4.2	1	17,560	146,33	2,579,125	49,021	2,530,103
	100	23,618	196,82	3,468,893	65,933	3,402,960
	500	19,144	159,53	2,811,775	53,443	2,758,331
	1000	16,456	137,13	2,416,975	45,939	2,371,035
การทดลองที่ 5.1	1	6,880	57.33	174,141	25,582	148,558
	100	15,024	125.20	380,252	56,054	324,197
	500	13,259	110.49	335,613	49,210	286,402
	1000	10,181	84.84	258,090	37,293	220,796

การทดสอบที่	Load Level	Total Request (Hit)	Output (RPS)	Average Bandwidth (bytes/sec)	Average rate of sent bytes (bytes/sec)	Average rate of receive bytes (bytes/sec)
การทดลองที่ 5.2	1	12,499	104.16	301,766	46,607	255,159
	100	14,747	122.89	356,037	55,018	301,018
	500	7,942	66.18	191,783	29,316	162,466
	1000	5,503	45.86	132,951	19,790	113,161
การทดลองที่ 6.1	1	1,867	15.56	393,529	6,418	387,110
	100	2,670	22.25	562,778	9,239	553,538
	500	3,131	26.09	659,973	10,649	649,323
	1000	3,772	31.43	795,127	12,553	782,574
การทดลองที่ 6.2	1	3,170	26.42	667,120.08	11,007.83	656,112.25
	100	2,955	24.62	621,875.12	10,251.75	611,623.38
	500	2,933	24.44	617,275.41	9,964.38	607,311.02
	1000	2,986	24.88	628,478.82	9,800.77	618,678.05

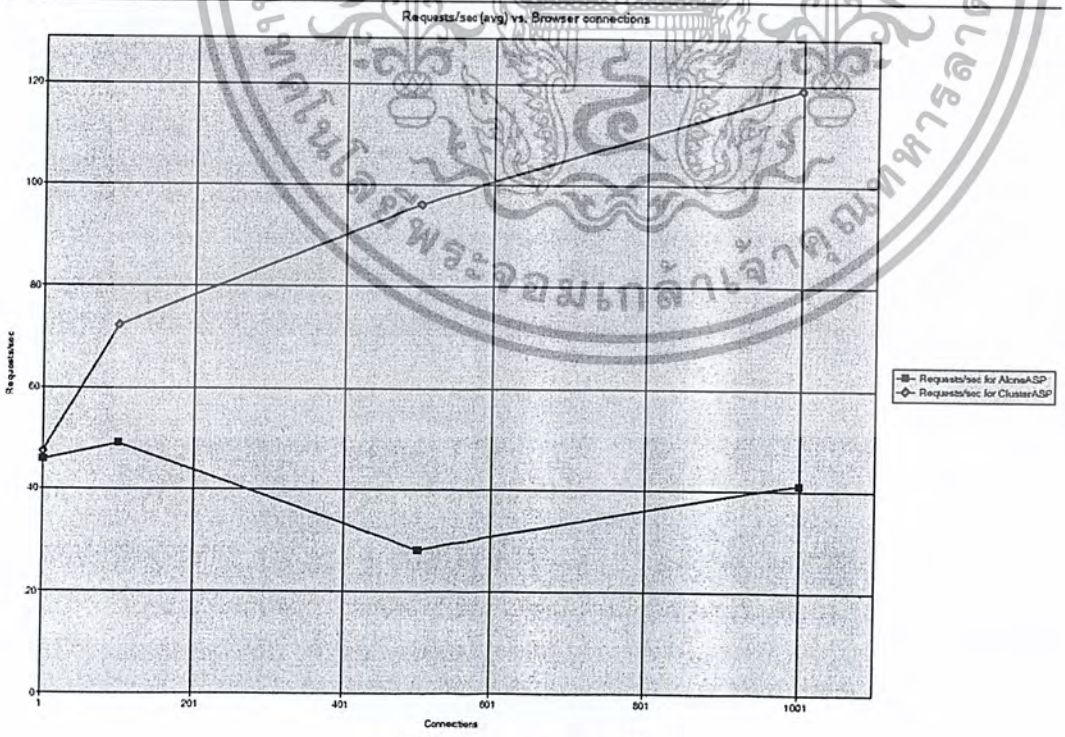
ตารางที่ 6-1 ผลการทดลองการใช้งานเน็ตเวิร์คโหลดบาลานซิ่ง

กราฟจากการทดลองที่ 1



รูปที่ 6-1 ผลการทดลองที่ 1

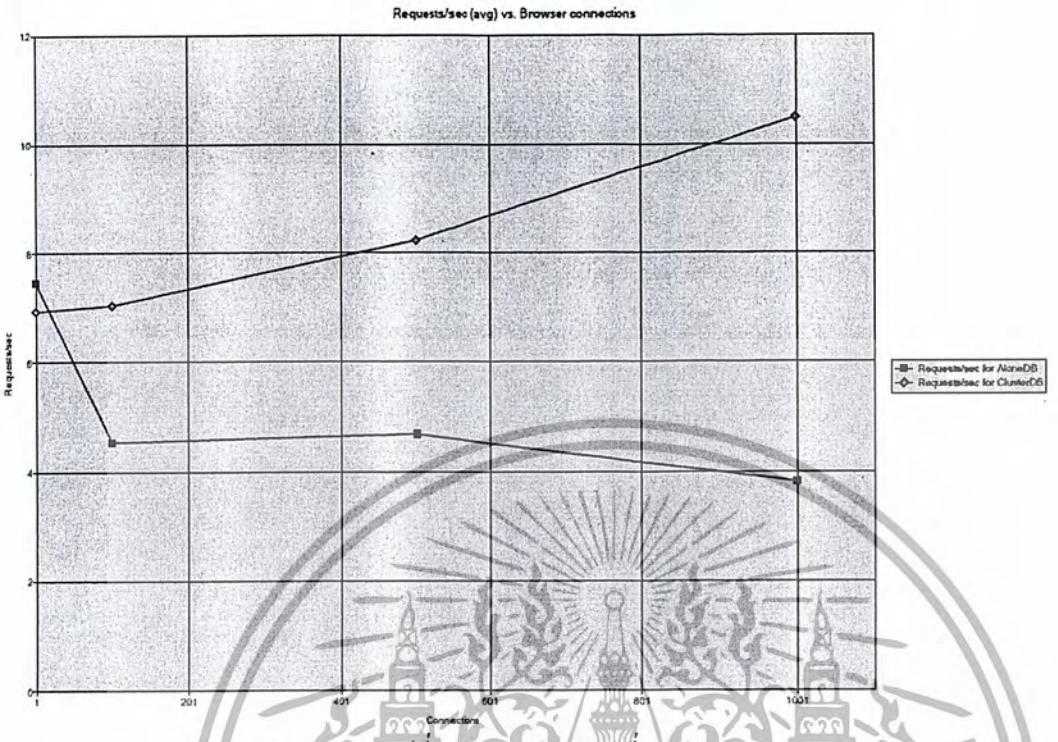
กราฟจากการทดลองที่ 2



รูปที่ 6-2 ผลการทดลองที่ 2

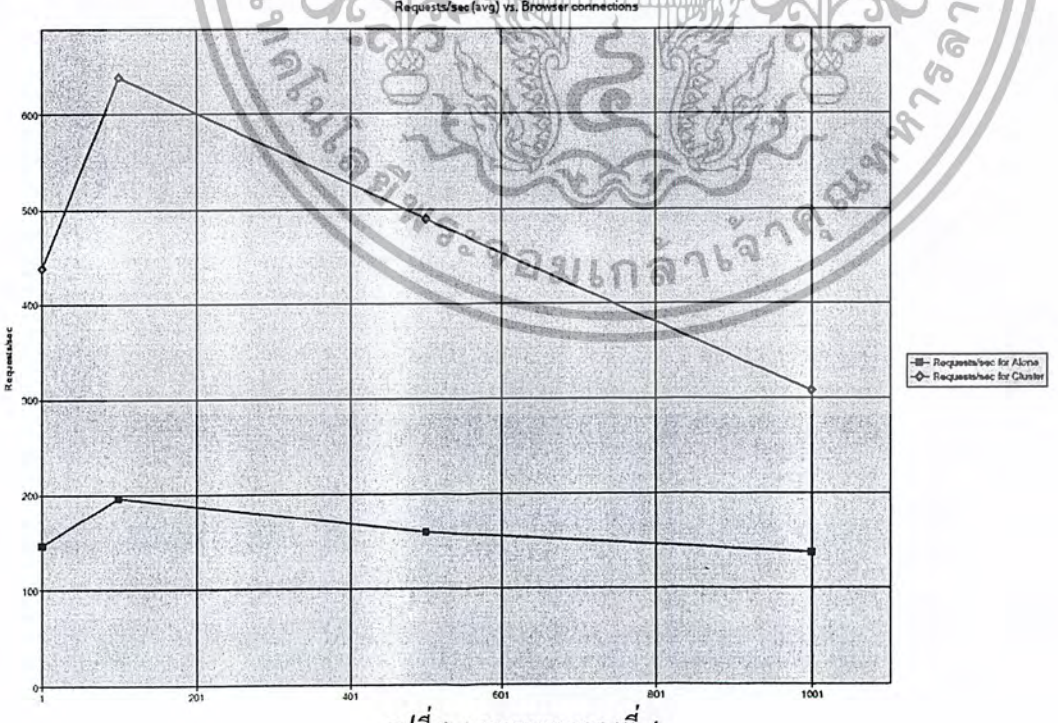
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กราฟจากการทดลองที่ 3



รูปที่ 6-3 ผลการทดลองที่ 3

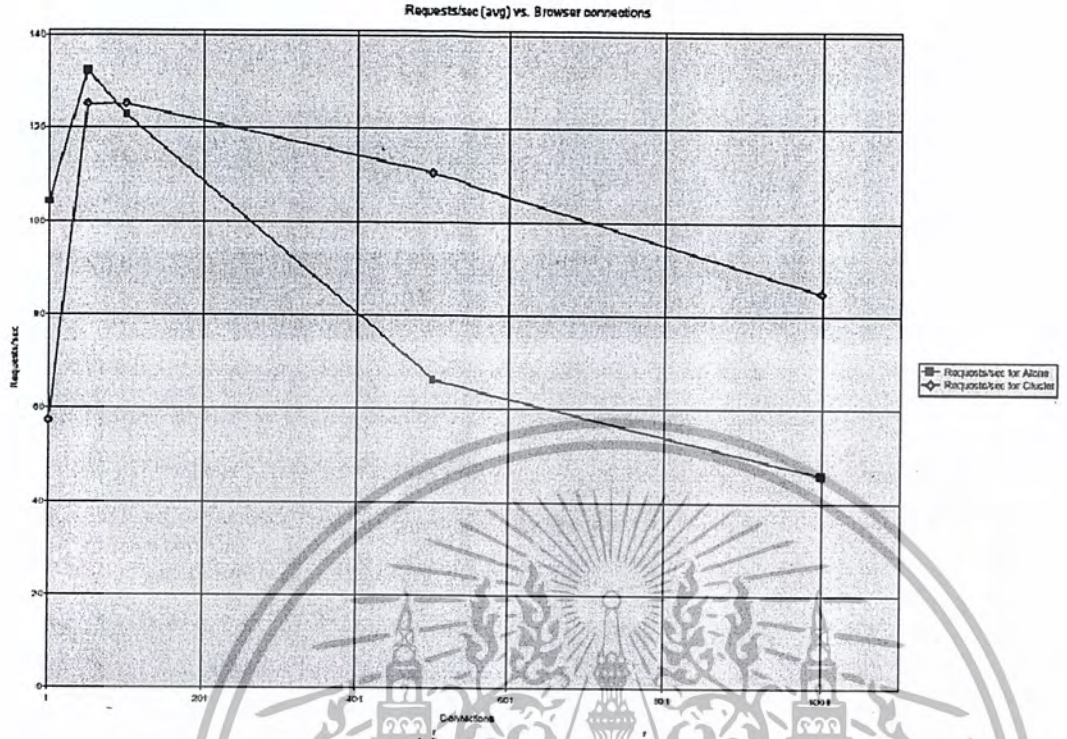
กราฟจากการทดลองที่ 4



รูปที่ 6-4 ผลการทดลองที่ 4

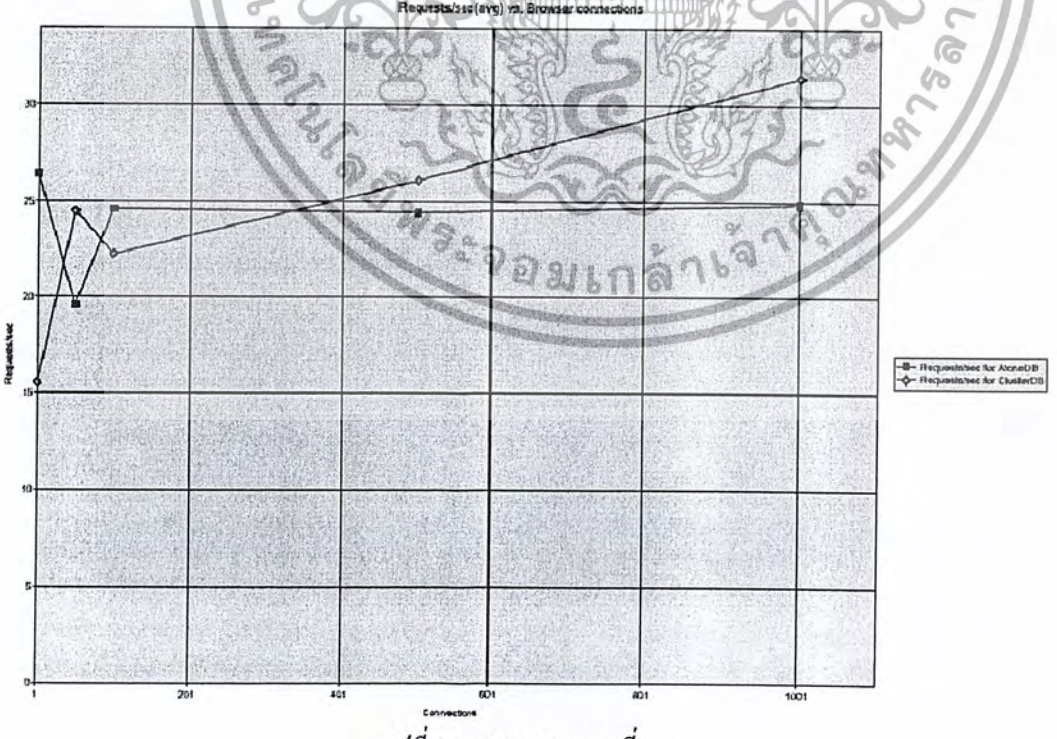
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กราฟจากการทดลองที่ 5



รูปที่ 6-5 ผลการทดลองที่ 5

กราฟจากการทดลองที่ 6



รูปที่ 6-6 ผลการทดลองที่ 6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7
สรุปผลการทดลองและเปรียบเทียบ

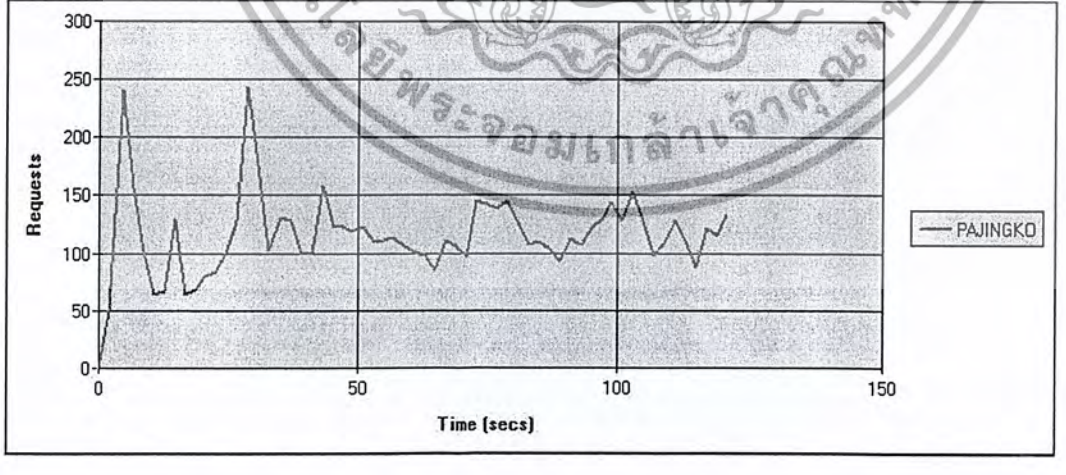
7.1 สรุปผลการทดลอง

จากผลการทดลอง จะเห็นได้ว่าเน็ตเวิร์คโหลดบาลานซิ่งสามารถช่วยเพิ่มประสิทธิภาพการทำงานให้กับเครือข่าย ทำให้สามารถรองรับการทำงานได้มากขึ้น และถ้าหากมีเครื่องที่อยู่ในคลัสเตอร์มากเท่าไหร่ เน็ตเวิร์คโหลดบาลานซิ่งก็จะมีประสิทธิภาพและสามารถรองรับการทำงานได้มากขึ้นเท่านั้น เน็ตเวิร์คโหลดบาลานซิ่ง สามารถทำงานได้ทั้งฝั่งของไมโครซอฟท์และฝั่งของจาวา โดยเมื่อดูจากผลการทดลองแล้วจะเห็นด้วยว่าโหลดบาลานซิ่งนั้น สามารถช่วยให้ทำงานได้ดีขึ้นทั้ง 2 ฝั่ง

จากผลการทดลอง ถ้าค่าโหลดเลเวลมีค่ามากมาก จะเห็นได้ชัดเลยว่าเน็ตเวิร์คโหลดบาลานซิ่งนั้นทำงานได้ดีกว่า สามารถรองรับการทำงานได้ดีกว่าเครื่องที่ทำงานเป็นเซิร์ฟเวอร์เพียงเครื่องเดียว แต่ถ้าหากค่าโหลดเลเวลมีค่าน้อยๆ จะเห็นความแตกต่างของเซิร์ฟเวอร์ที่เป็นเครื่องเดียว กับเซิร์ฟเวอร์ที่เป็นคลัสเตอร์น้อยมาก หรืออาจจะไม่แตกต่างเลย เนื่องจากข้อมูลที่ส่งเข้ามานั้น มีค่าน้อย เมื่อเกิดการแบ่งขึ้น การทำงานก็จะตกไปอยู่กับเครื่องใดเครื่องหนึ่งในคลัสเตอร์เท่านั้น ทำให้ไม่มีผลต่อการเช็คประสิทธิภาพของระบบมากนัก

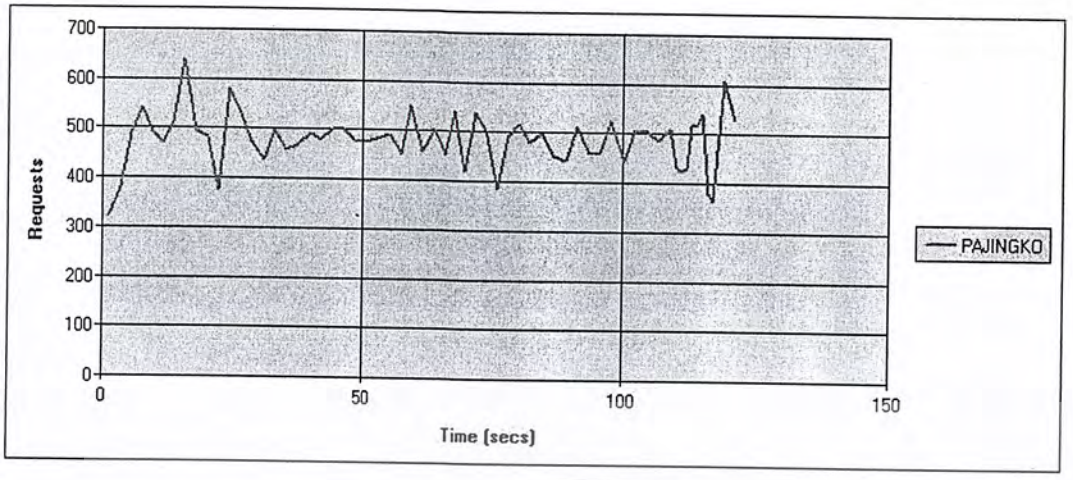
7.2 เปรียบเทียบประสิทธิภาพ

จากผลการทดลองการวัดประสิทธิภาพระหว่างฝั่งไมโครซอฟท์และฝั่งจาวา จะเห็นได้ว่าการเน็ตเวิร์คโหลดบาลานซิ่งนั้นทำการแบ่งการทำงานได้เช่นเดียวกัน แต่ทางฝั่งจาวาจะสามารถทำงานได้อย่างมีประสิทธิภาพมากกว่า จากรูป เป็นการเปรียบเทียบการทำโหลดบาลานซิ่งระหว่างฝั่งไมโครซอฟท์และฝั่งจาวา โดยใช้สภาวะแวดล้อมต่างๆเหมือนกัน



รูปที่ 7-1 ผลการทดสอบการทำงานของโหลดบาลานซิ่ง ของฝั่งไมโครซอฟท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7-2 ผลการทดสอบการทำงานของโหลดบาลานซิ่งของฝั่งจาวา

จากรูปทั้งสองจะเห็นได้ว่า การทำงานของทางฝั่งจาวาจะดีกว่ามากเนื่องจากการแบ่งงานของฝั่งจาวานั้นทำได้อย่างสมบูรณ์แบบมากกว่า สามารถจัดการให้มีหลายเซิร์ฟเวอร์ในเครื่องเดียวได้ โดยแบ่งการทำงานของแต่ละเซิร์ฟเวอร์โดยใช้พอร์ตที่รับเข้ามาเป็นตัวแบ่งการทำงานทำให้จาวานั้นใช้ทรัพยากรของเครื่องได้คุ้มค่ากว่า ทางฝั่งของไมโครซอฟท์นั้น ใช้ได้เพียงเครื่องละ 1 เซิร์ฟเวอร์เท่านั้น จาวาจึงมีประสิทธิภาพในการทำงานเน็ตเวิร์คโหลดบาลานซิ่งได้เหนือกว่าของไมโครซอฟท์

7.3 ปัญหาที่เกิดขึ้นระหว่างการทดลอง

1) เครื่องที่ใช้ทดสอบการทำงานของเน็ตเวิร์คโหลดบาลานซิ่งนั้น ต้องเป็นเครื่องที่มีประสิทธิภาพพอสมควรเนื่องจากหากเป็นเครื่องที่สเปคไม่ดีก็จะไม่สามารถแสดงค่าที่ถูกต้องออกมาได้
2) การติดตั้งค่าแต่ละค่าในเน็ตเวิร์คโหลดบาลานซิ่งนั้น ต้องมีความละเอียดพอสมควรเนื่องจากหากเราตั้งค่าผิดไป ค่าที่ออกมาอาจจะไม่ได้เลยก็ได้ หรืออาจทำให้เกิดปัญหาตามมาได้ และต้องคำนึงถึงการใช้งานเป็นหลักอีกด้วย
3) ค่าที่ออกมานั้น อาจเกิดการผิดพลาดได้เนื่องจากใช้ฮับเป็นอุปกรณ์ติดต่อ อาจมีสัญญาณ บรอดแคสท์ที่มาจากเครื่องอื่นมารบกวนการทำงานของเครื่องที่เราสนใจได้ เพราะฉะนั้นเราควรทดสอบเมื่อมีการใช้งานของฮับน้อยกว่า จะทำให้ค่าที่ออกมาแสดงความเป็นจริงได้มากขึ้น
4) อุปกรณ์เน็ตเวิร์คบางชนิดไม่สามารถทำงานกับเน็ตเวิร์คโหลดบาลานซิ่ง เพราะฉะนั้นเราควรตรวจสอบก่อนว่าอุปกรณ์เน็ตเวิร์คที่เราใช้นั้นสามารถทำงานร่วมกับเน็ตเวิร์คโหลด บาลานซิ่งได้หรือไม่ ก่อนการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
บทวิจารณ์ และสรุป

8.1 สรุปผลการดำเนินงาน

ผลดำเนินงานติดตั้งและทดสอบมีผลการทดสอบที่น่าพอใจ โดยเว็บโลจิกมีประสิทธิภาพการทำงานที่ดีกว่าของไมโครซอฟท์ เนื่องจากใช้ทรัพยากรที่มีอยู่อย่างเต็มที่โดยใช้ปริมาณเซิร์ฟเวอร์มากกว่าในเครื่องเซิร์ฟเวอร์เครื่องหนึ่งของไมโครซอฟท์ซึ่งสามารถใช้ได้แค่ IIS เซิร์ฟเวอร์เพียงตัวเดียว ซึ่งโครงงานชิ้นนี้นำเสนอถึงการพัฒนาโหลดบาลานซิ่งให้มีประสิทธิภาพสูงสุดโดยทรัพยากรที่มีนั้นจำกัด

สรุปการทำโหลดบาลานซิ่งของไมโครซอฟท์และเว็บโลจิกนั้น ไมโครซอฟท์ถ้าจะพัฒนาให้รองรับปริมาณงานที่มากขึ้นนั้นต้องลงทุนสูงกว่าเว็บโลจิกและมีประสิทธิภาพที่ไม่มากเท่าทีควรแต่มีข้อดีในด้านการรองรับความผิดพลาดที่เกิดขึ้นได้อย่างมีประสิทธิภาพซึ่งเว็บโลจิกเองทำได้ไม่สมบูรณ์นักเนื่องจากว่าต้องพึ่งโหลดบาลานเซอร์ เช่น IIS, apache หรือ ตัวเว็บโลจิกเอง ซึ่งการทำโหลดบาลานซิ่งวิธีนี้ยังไม่ดีพอเนื่องจากใช้ได้แค่วิธี Round-robin และถ้าใช้โหลดบาลานเซอร์แบบฮาร์ดแวร์ก็จะต้องลงทุนเพิ่มอีก

การทำงานตามทฤษฎีนั้นในส่วนของไมโครซอฟท์ไม่สามารถคาดเดาได้โดยง่ายเพราะใช้วิธีการสุ่ม ส่วนของเว็บโลจิกนั้นสามารถสังเกตดูได้ค่อนค้างชัดเจนซึ่งจากการทดสอบทั้งคู่สามารถจัดการกับความผิดพลาดได้เป็นอย่างดีแต่เว็บโลจิกจะมีปัญหาอยู่บ้างในการเพิ่มเซิร์ฟเวอร์เข้าคลัสเตอร์ ซึ่งต้องติดตั้งเว็บแอพพลิเคชันใหม่ ส่วนไมโครซอฟท์ก็ต้องทำเช่นกันแต่โอกาสผิดพลาดน้อยกว่าเนื่องจากเว็บโลจิกจะมีการประกาศออบเจ็คซึ่งอาจเกิดข้อผิดพลาดได้

8.2 แนวทางการพัฒนาต่อ

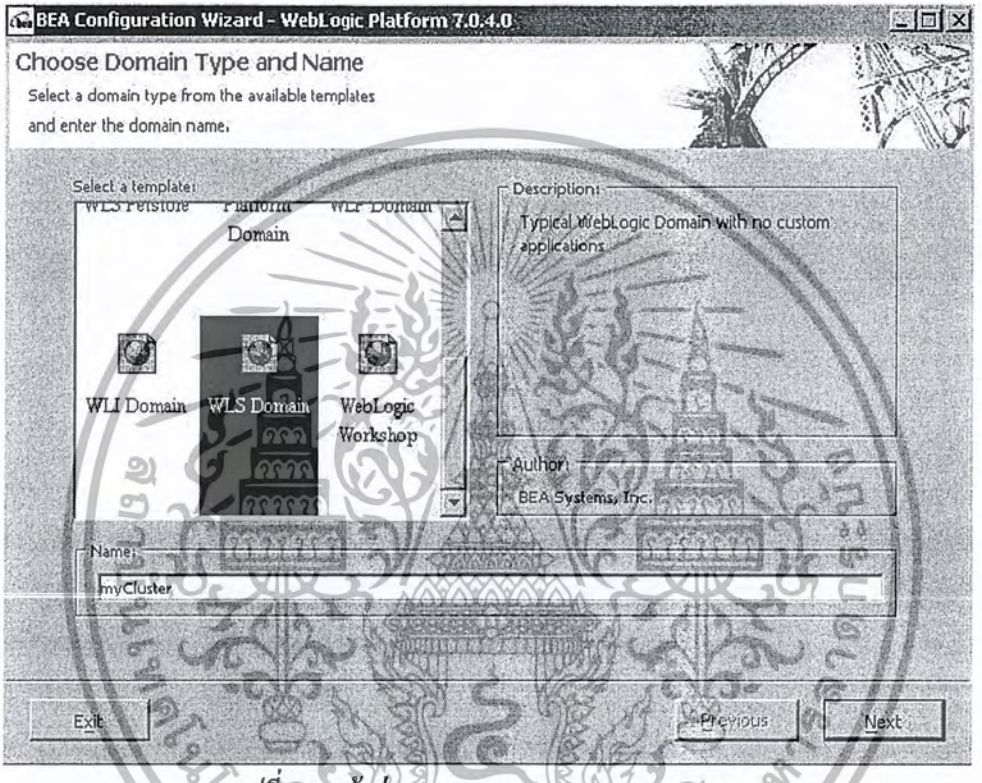
ถึงแม้ว่า การทำงานของเว็บเซอร์วิสนั้นสามารถประยุกต์ให้ทำงานโดยใช้เน็ตเวิร์คโหลดบาลานซิ่งได้แต่ก็ยังมีบางจุดที่เน็ตเวิร์คโหลดบาลานซิ่งไม่สามารถช่วยเพิ่มประสิทธิภาพได้ ยกตัวอย่างเช่น เรื่องของคาด้าเบสเซิร์ฟเวอร์ เราไม่สามารถใช้เน็ตเวิร์คโหลดบาลานซิ่งได้ แต่ยังมีอีกวิธีหนึ่งที่จะใช้เพื่อเพิ่มประสิทธิภาพการทำงานให้กับดาด้าเบสเซิร์ฟเวอร์ได้ คือวิธี คลัสเตอร์ ซึ่งเป็นอีกวิธีที่ช่วยเพิ่มประสิทธิภาพได้ และเราอาจจะเพิ่มความสามารถให้กับเซิร์ฟเวอร์อีกก็ได้ เช่นการรักษาความปลอดภัยระหว่างการส่งข้อมูลในแต่ละเซิร์ฟเวอร์เพื่อให้มั่นใจว่าข้อมูลที่ส่งโดยผ่านเว็บเซอร์วิสนั้นปลอดภัยและถูกต้อง โดยการพัฒนาความปลอดภัยและโหลดบาลานซิ่งสามารถทำควบคู่กับได้จึงเหมาะแก่การนำไปใช้งานจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก
ขั้นตอนการติดตั้งเว็บโลจิกคลัสเตอร์

1. ขั้นตอนการติดตั้งเว็บโลจิกคลัสเตอร์

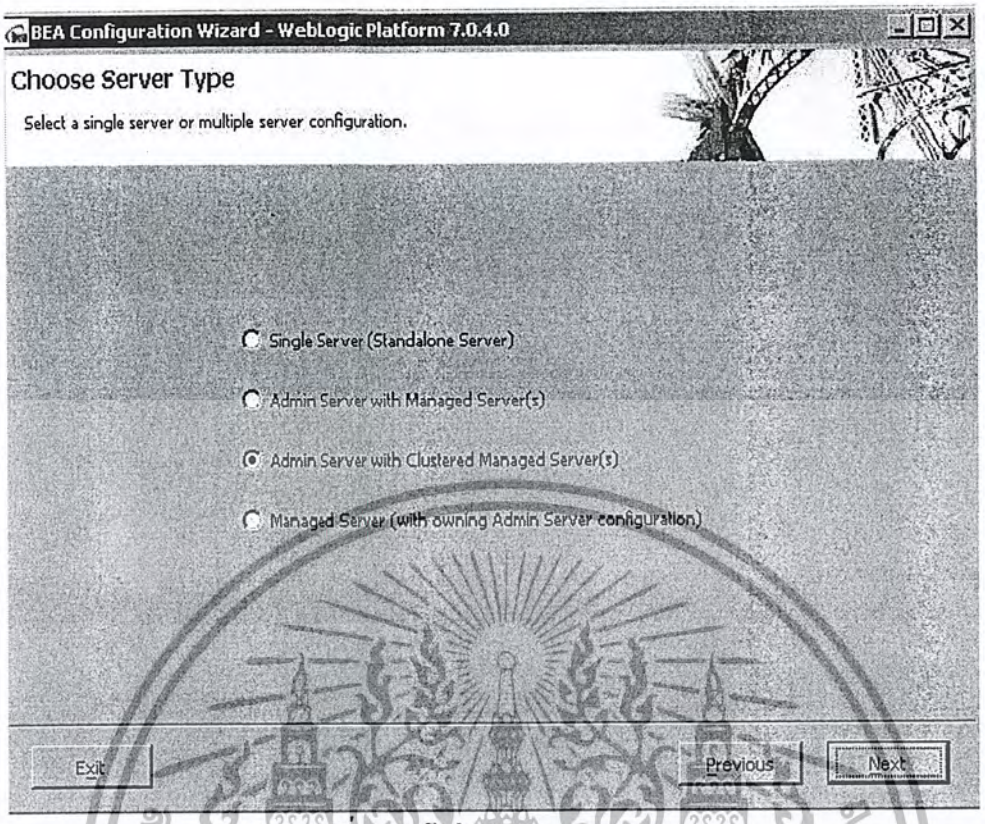
1. ไปที่ Start → Programs → BEA WebLogic Platform → Domain Configuration Wizard



รูปที่ ก-1 หน้าต่าง Choose Domain Type and Name

2. ในหน้าต่าง Choose Domain Type and Name ให้เลือกประเภทโดเมนเป็น WLS Domain แล้วตั้งชื่อโดเมน แล้วคลิกเน็ค (Next)
3. ในหน้าต่าง Choose Server Type ให้เลือก Admin Server with Clustered Managed Server(s) แล้วคลิกเน็ค

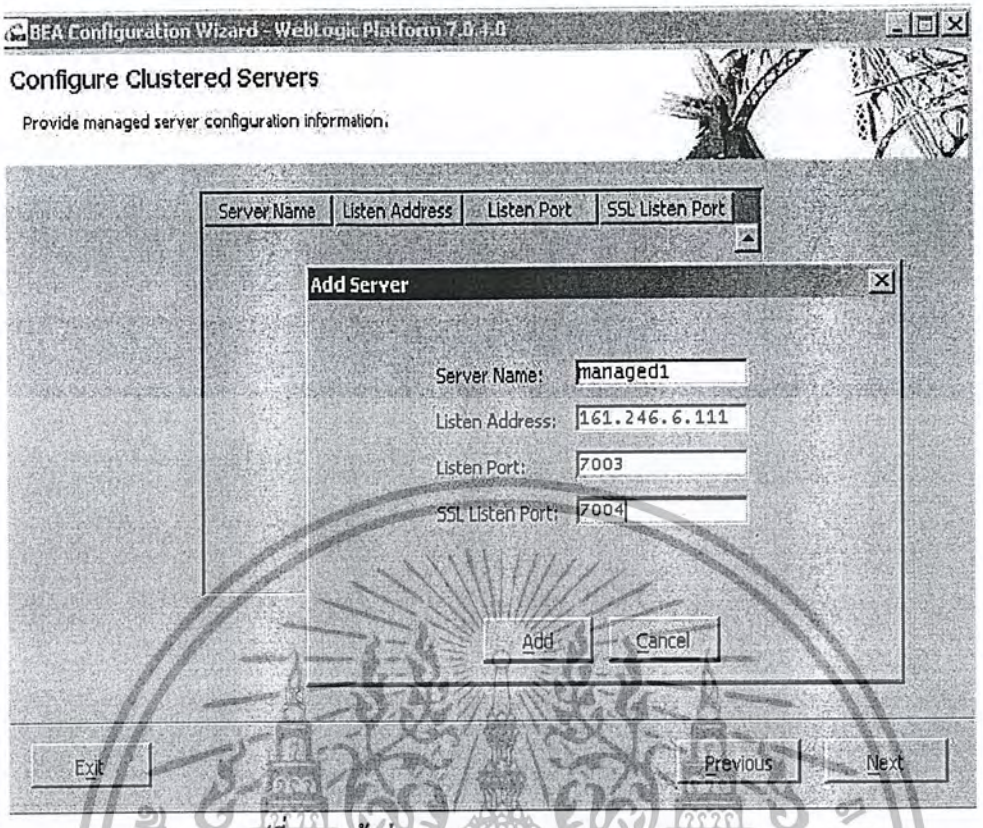
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ก-2 หน้าต่าง Choose Server Type

4. ในหน้าต่าง Choose Domain Location window ให้เลือกไดเล็กตอรี่ที่จะเก็บข้อมูล แต่ต้องแน่ใจว่าอยู่ใน*\BEA_HOME* โดย *BEA_HOME* เป็นไดเล็กตอรี่ที่ติดตั้ง WebLogic Platform
5. ในหน้าต่าง Configure Clustered Server(s) คลิกแอด(Add) เพื่อทำการตั้งค่า Managed Server สำหรับคลัสเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



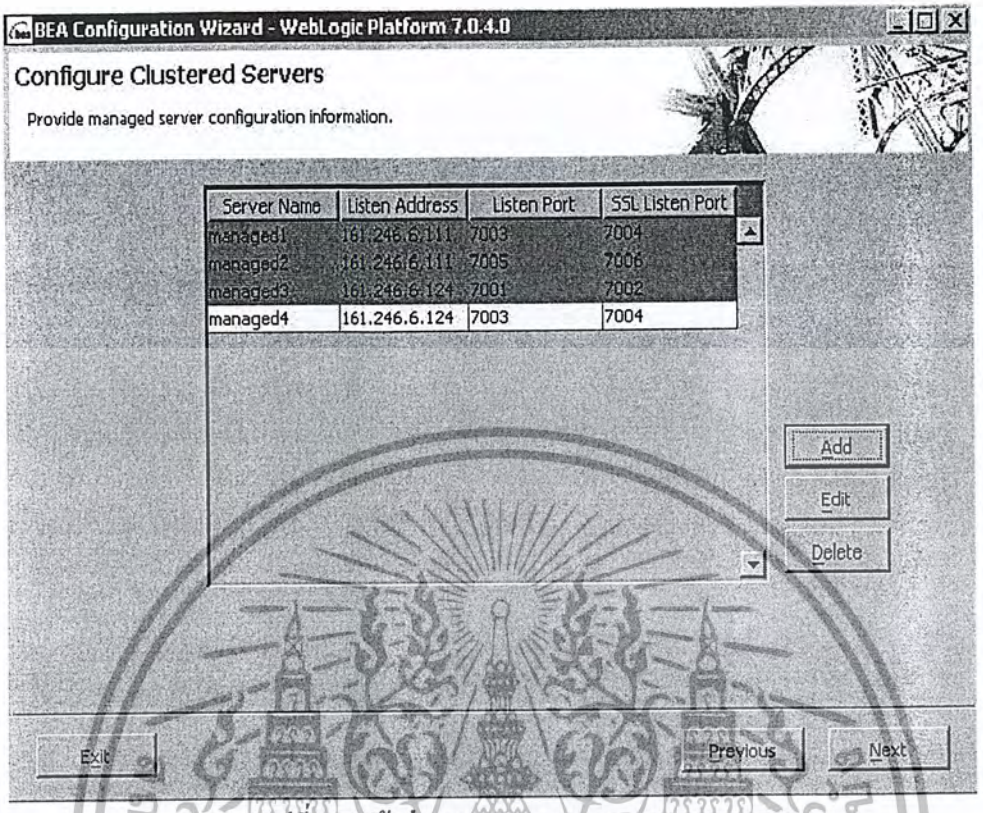
รูปที่ ก-3 หน้าต่าง Choose Domain Type and Name

6. ในหน้าต่าง Add Server ให้ใส่

- Server Name—ให้ใส่ชื่อเซิร์ฟเวอร์ที่ไม่ซ้ำในแต่ละเซิร์ฟเวอร์
- Server Listen Address—ใส่ IP address หรือ ชื่อเครื่อง
- Server Listen Port—ใส่ค่าพอร์ตที่จะใช้ ตั้งแต่ 1 ถึง 65535

คลิกแอด เพื่อเพิ่มข้อมูล

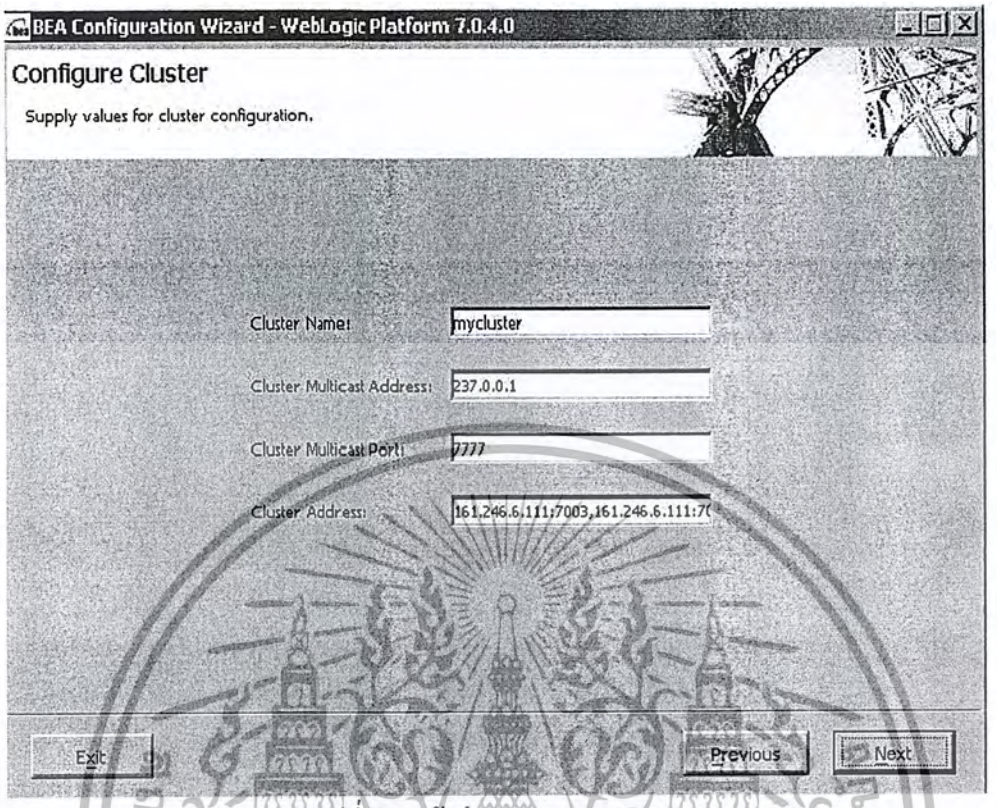
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ก-4 หน้าต่าง Configure Clustered Servers

7. ในหน้าต่าง Add Server ให้เพิ่ม Managed Server ไปยังคลัสเตอร์ โดยทำซ้ำตามจำนวนที่ต้องการ
8. ในหน้าต่าง Configure Cluster ให้ใส่
   - Cluster Name—จะมีค่าตั้งต้นคือ *mycluster* ให้เปลี่ยนชื่อตามที่ต้องการ
   - Cluster Multicast Address – จะมีค่าตั้งต้นคือ 237.0.0.1 สามารถตั้งค่าได้ตั้งแต่ 224.0.0.0 ถึง 239.255.255.255
   - Cluster Multicast Port – จะมีค่าตั้งต้นคือ 7777
   - Cluster Address – เป็นค่า IP Address และ port ของเซิร์ฟเวอร์ ทั้งหมดที่มีในคลัสเตอร์ คลิกเน็ค

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



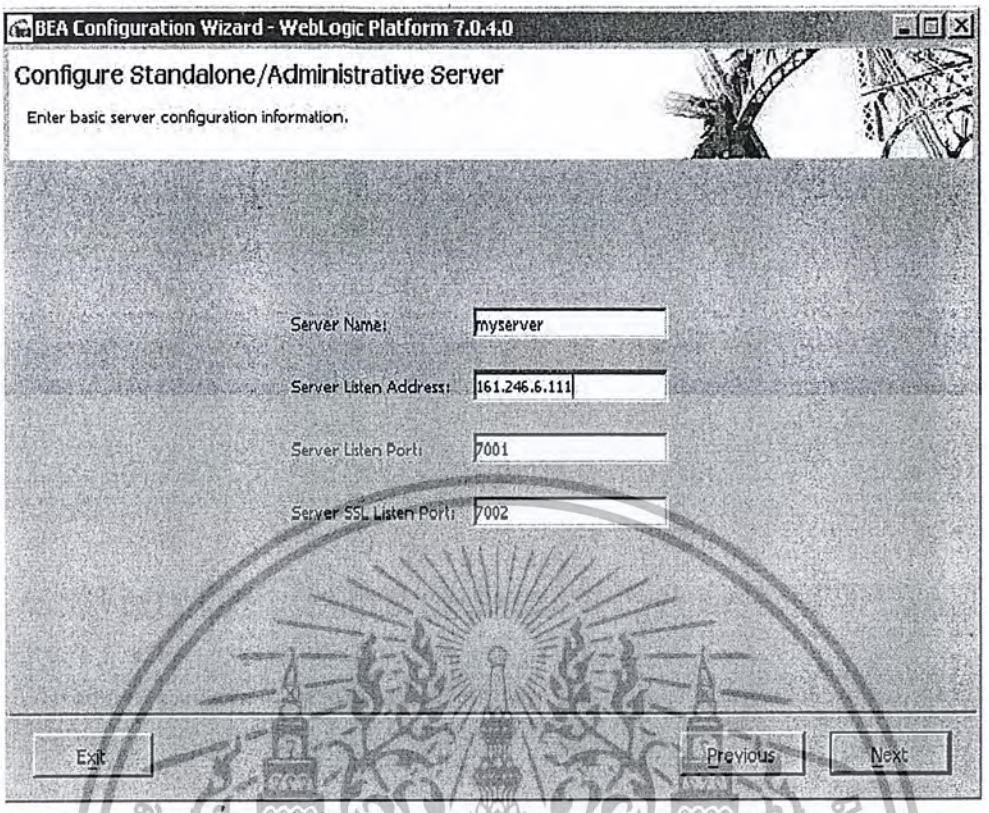
รูปที่ ก-5 หน้าต่าง Configure Cluster

9. ในหน้าต่าง Configure Admin Server (with Cluster) ให้ใส่

- Server Name – จะมีค่าตั้งต้นคือ *myserver* จะเป็นชื่อของเซิร์ฟเวอร์ที่ใช้ดูแลเซิร์ฟเวอร์อื่นๆ
- Server Listen Address – ให้ใส่ IP Address หรือชื่อเครื่องที่ติดตั้ง
- Server Listen Port – จะมีค่าตั้งต้นคือ 7001 สามารถตั้งค่าได้ตั้งแต่ 1 ถึง 65535

คลิกเน็ค

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ก-6หน้าต่าง Configure Standalone/Administrative Server

10. ในหน้าต่าง System User Name and Password ให้ใส่ชื่อและรหัสที่ใช้ในการเปิด ปิดและดูแล
11. ในหน้าต่าง Install Server as Windows Service
    - เลือก Yes เพื่อติดตั้งโดเมนในวินโดว์เซอร์วิสเพื่อให้เซิร์ฟเวอร์เริ่มอัตโนมัติเมื่อเปิดวินโดว์
    - เลือก No ถ้าไม่ต้องการติดตั้งในวินโดว์เซอร์วิส

    คลิกเน็ค
12. ในหน้าต่าง Install Domain
    - เลือก Yes สำหรับเพิ่มเมนูสตาร์ทโดเมนในวินโดว์สตาร์ทเมนู
    - เลือก No สำหรับเพิ่มเมนูสตาร์ทโดเมนในวินโดว์สตาร์ทเมนู

    คลิกเน็ค
13. ในหน้าต่าง Configuration Summary จะเสดงข้อมูลในการสร้างทั้งหมด คลิก Previous เพื่อแก้ไขข้อมูล หรือ Create เพื่อสร้างโดเมน
14. ในหน้าต่าง Configuration Wizard Complete เลือก End Configuration Wizard เพื่อออก
15. ทำการติดตั้งเซิร์ฟเวอร์ที่จะต่อเข้าคลัสเตอร์โดยเปลี่ยนในขั้นตอนที่ 2 โดยเลือกเป็น Managed Server(with owning Admin Server configuration) แล้วทำตามขั้นตอนข้างต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. การติดตั้ง Proxy Plug-ins บน IIS 5.0

การติดตั้ง Proxy Plug-ins เพื่อทำโหลดบาลานซิ่งการใช้งาน HTTP โดยใช้วิธี Round-robin เพื่อกระจายงานให้เว็บโลจิกคลัสเตอร์อีกที โดยมรวิธีการติดตั้งดังนี้

1. คัดลอกไฟล์ *iisproxy.dll* จากไดเล็กตอรี่ */bin* ของเว็บโลจิกเซิร์ฟเวอร์ไปไว้ในที่ที่ IIS เซิร์ฟเวอร์สามารถเรียกใช้งานได้โดยต้องสร้างไฟล์ *iisproxy.ini* ด้วย
2. เริ่ม IIS Internet Service Manager โดยเลือกจาก Administrative Tools
3. เลือกเว็บไซต์ที่ต้องการ (ปกติจะเป็น Default Web Site)
4. ให้เริ่มการทำงานเว็บไซต์นั้น
5. เปิดหน้าต่างพรอบเพอตี้(Properties) โดยการคลิกขวาที่ชื่อเว็บนั้น
6. ในหน้าต่างพรอบเพอตี้ให้เลือกแถบ home directory และคลิกที่ปุ่ม Configuration ในส่วนของ Application Setting
7. เลือก Configure Proxying by File Extension
8. ในแถบ App Mapping คลิก แอดเพื่อเพิ่มประเภทไฟล์และดำการตั้งค่าเพื่อส่งต่อไปยังเว็บโลจิกเซิร์ฟเวอร์
9. ใน dialog box ให้เลือกไปยังไดเล็กตอรี่ที่เก็บไฟล์ *iisproxy.dll*
10. ตั้งค่านามสกุลที่ต้องการให้เว็บโลจิกเซิร์ฟเวอร์ทำงาน เช่น .jsp หรือ .html ฯลฯ
11. ไม่ต้องเลือกในช่อง Check That File Exists check box
12. เมื่อเสร็จแล้วให้คลิดโอเคเพื่อเก็บข้อมูลแล้ววนซ้ำในแต่ละประเภทไฟล์ที่ต้องการ
13. สร้างไฟล์ *iisproxy.ini* และเก็บไฟล์ในที่เดียวกับไฟล์ *iisproxy.dll* ตัวอย่างของไฟล์ *iisproxy.ini* เป็นดังนี้

```
WebLogicCluster=161.246.6.127:7001,161.246.6.127:7003,161.246.6.124:7001,
                161.246.6.124:7003
ConnectTimeoutSecs=20
ConnectRetrySecs=2
Debug=on
```

โดยใน WebLogicCluster จะบอกถึง IP Address ของเครื่องที่อยู่ในเว็บโลจิกคลัสเตอร์เซิร์ฟเวอร์

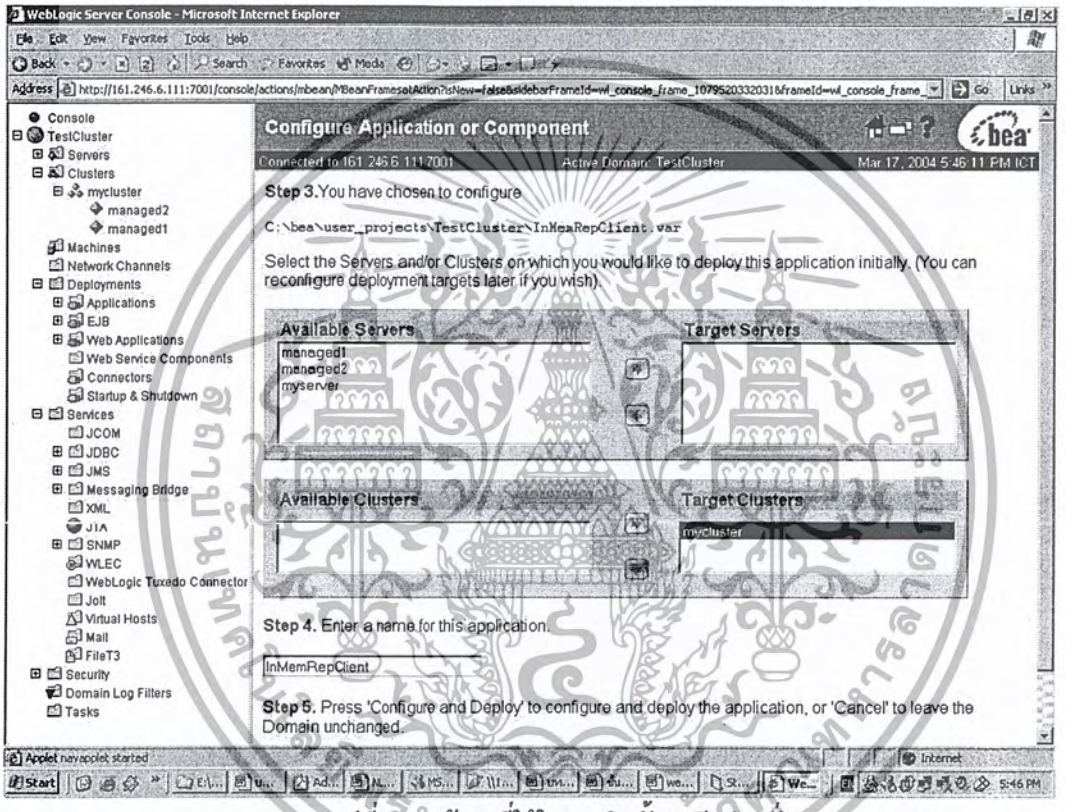
3. การติดตั้งแอปพลิเคชั่นบนคลัสเตอร์

โดยในตัวอย่างนี้จะทำหารติดตั้ง *InMemRepClient.war* ที่เว็บโลจิกมีมาให้โดยใช้คอนโซวร์ของเว็บโลจิก

1. เริ่มเว็บโลจิกคลัสเตอร์เซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

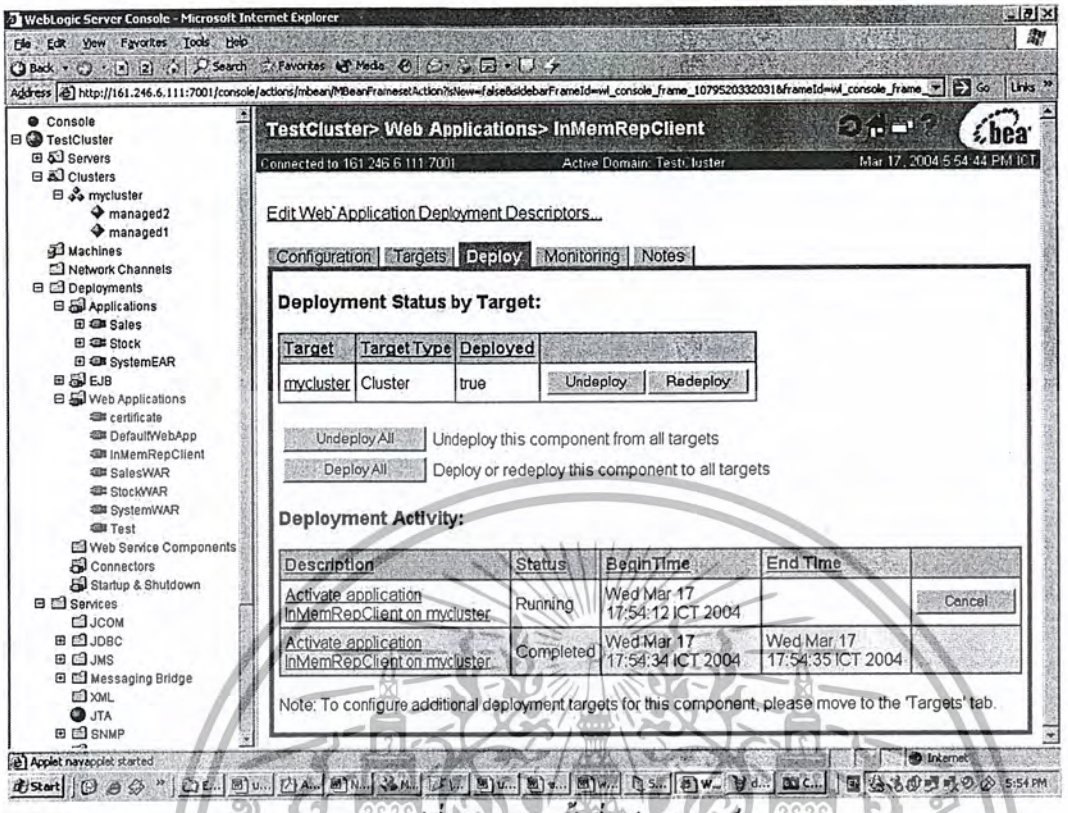
2. ไปที่คอนโซวร์โดยไปที่ *http://localhost:7001/console* โดยค่าอาจเปลี่ยนแปลงตามค่าที่เรากำหนด
3. คลิกที่ Web Applications
4. คลิกที่ Configure A New Web Application
5. เลือกไปที่แอปพลิเคชั่นที่ต้องการติดตั้ง
6. เลือกชื่อคลัสเตอร์ที่จะทำการติดตั้ง (mycluster) และคลิก Configure And Deploy ดังรูปที่ 5-9



รูปที่ ก-7 หน้าจอที่ใช้ในการติดตั้งแอปพลิเคชั่น

7. เมื่อติดตั้งเสร็จจะแสดงดังรูปที่ ก-8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ก-8 การติดตั้งที่เสร็จสมบูรณ์

4. การเข้าถึงแอปพลิเคชั่นผ่าน Proxy Plug-ins

โดยปกติเราจะเข้าถึงเว็บโลจิกเซิร์ฟเวอร์ได้โดยผ่าน URL เช่น

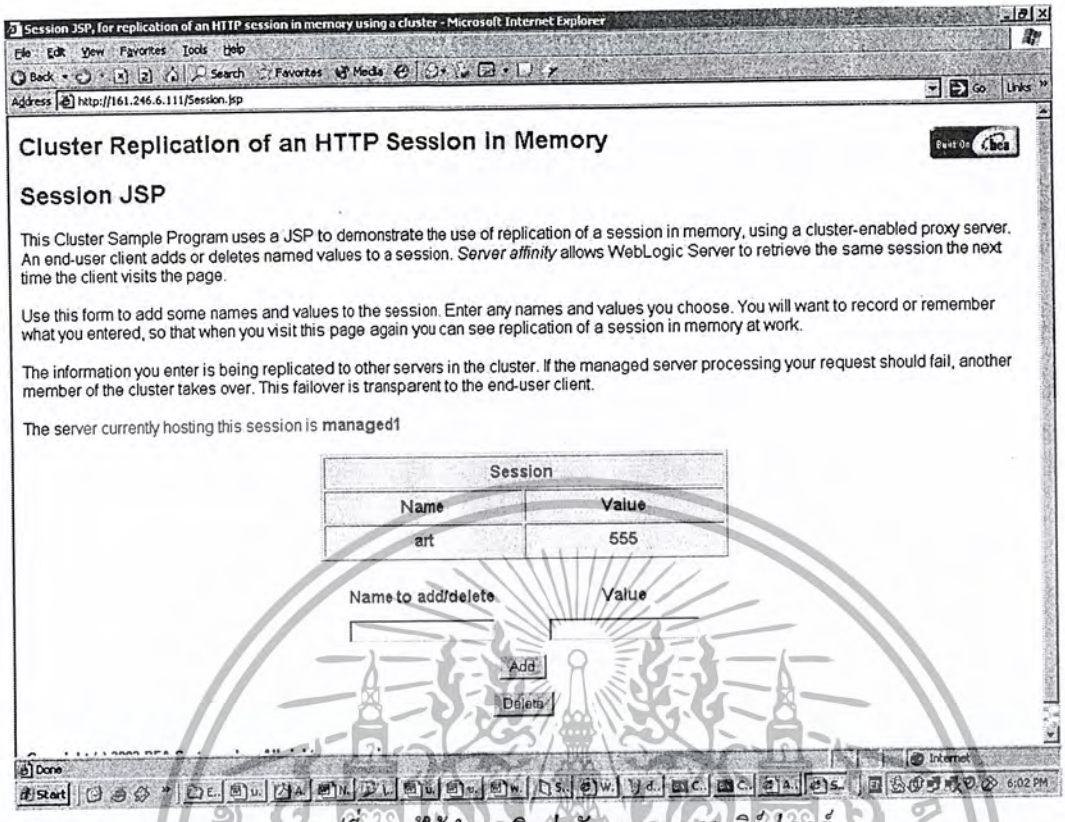
http://localhost:7001/InMemRepClient/Session.jsp

การใส่ URL เช่นนี้จะเรือไปที่เว็บโลจิกเซิร์ฟเวอร์โดยตรง แต่เมื่อทำการติดตั้ง Proxy Plug-ins เรียบร้อยจะสามารถติดต่อผ่าน URL นี้ได้

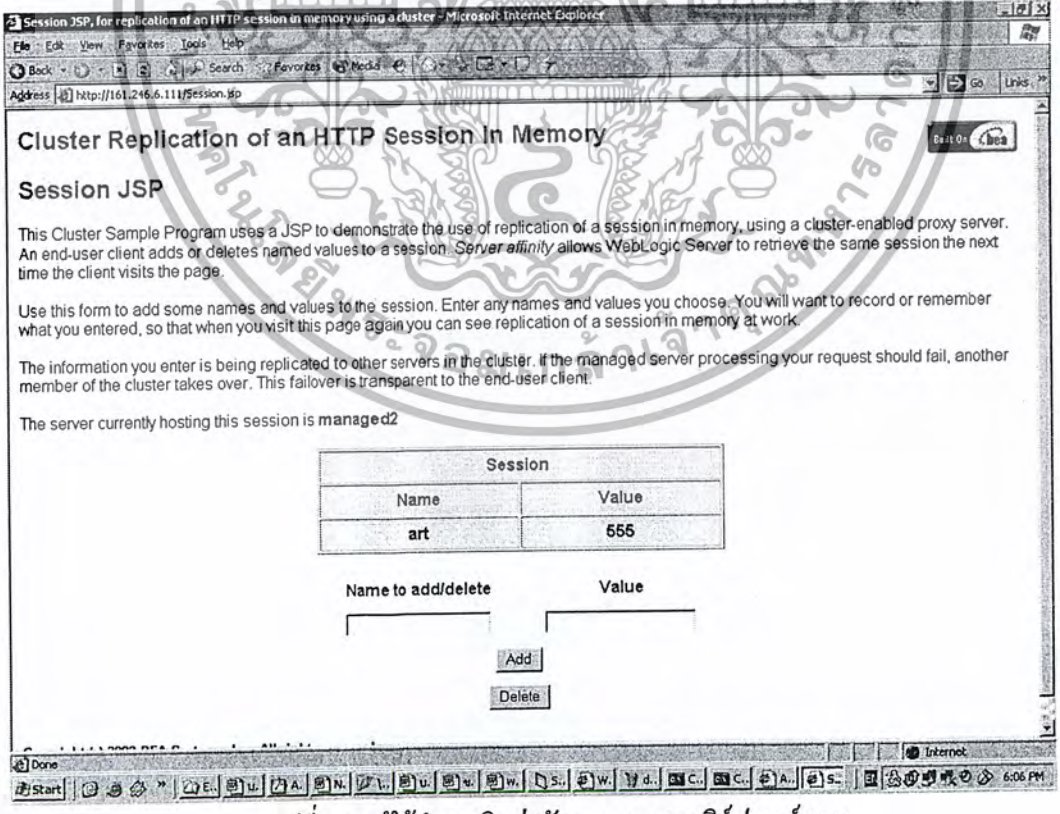
http://localhost/InMemRepClient/Session.jsp

ทำการทดสอบ Session ของระบบโดยดังรูปทำการติดต่ออยู่กับ *managed1* เซิร์ฟเวอร์ ให้ลองใส่ค่าดู จากนั้นทำการปิด *managed1* เซิร์ฟเวอร์จะเห็นได้ว่า *managed2* เซิร์ฟเวอร์เข้ามาทำงานแทนโดยค่าที่ใส่ยังถูกต้องครบถ้วน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ก-9 ผู้ใช้ทำการติดต่อกับ managed1 เซิร์ฟเวอร์



รูปที่ ก-10 ผู้ใช้ทำการติดต่อกับ managed2 เซิร์ฟเวอร์แทน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

ขั้นตอนการติดตั้งโหลดบาลานซิ่งบน

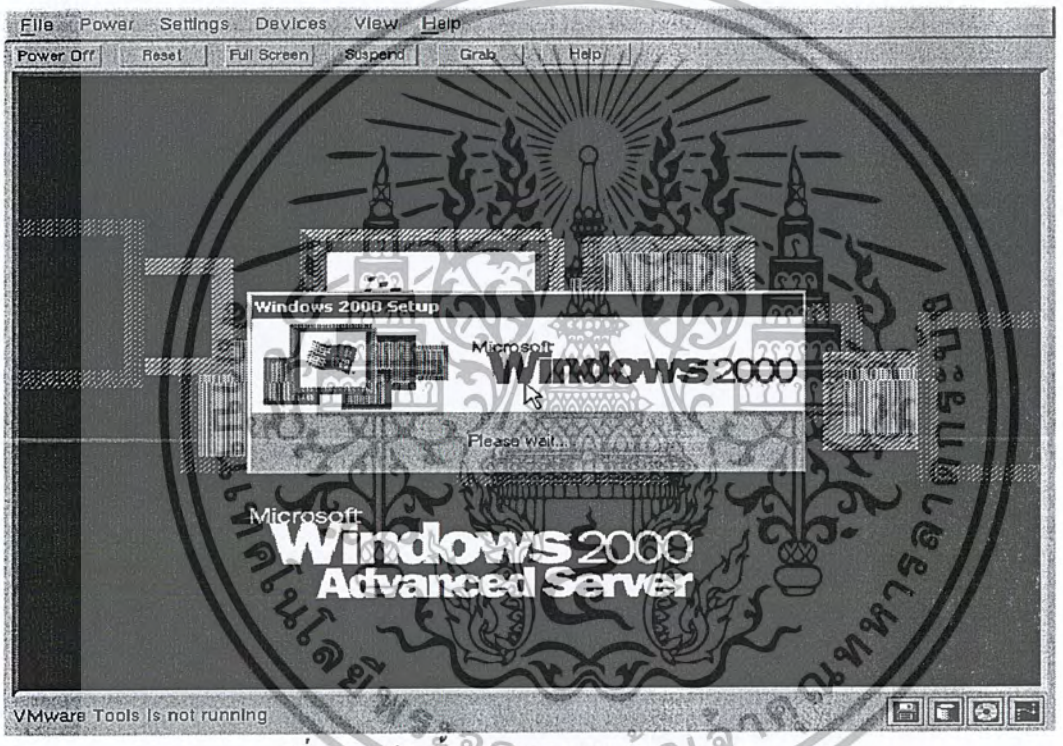
Microsoft Windows 2000 Advanced Server

1. ขั้นตอนการติดตั้ง

ในการติดตั้งซอฟท์แวร์ของแต่ละอุปกรณ์นั้น มีการติดตั้งโดยแยกตามแต่ละอุปกรณ์ได้ดังนี้

เซิร์ฟเวอร์ 1

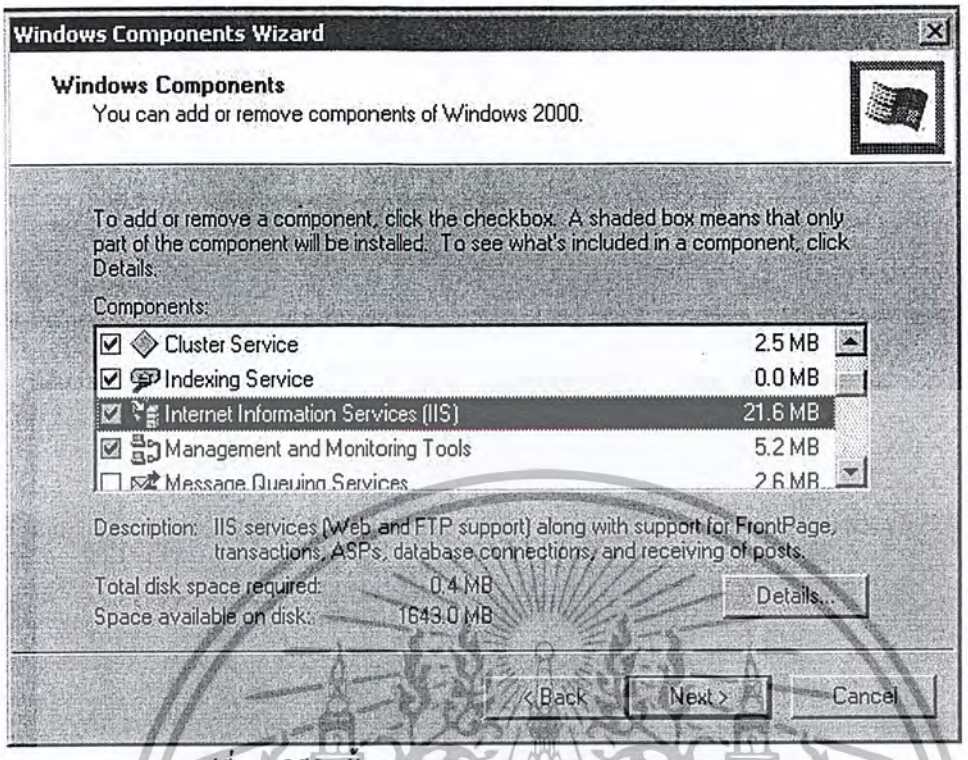
1. ติดตั้ง Windows 2000 Advanced Server ก่อน



รูปที่ ข --1 วิธีติดตั้ง Windows 2000 Advanced Server

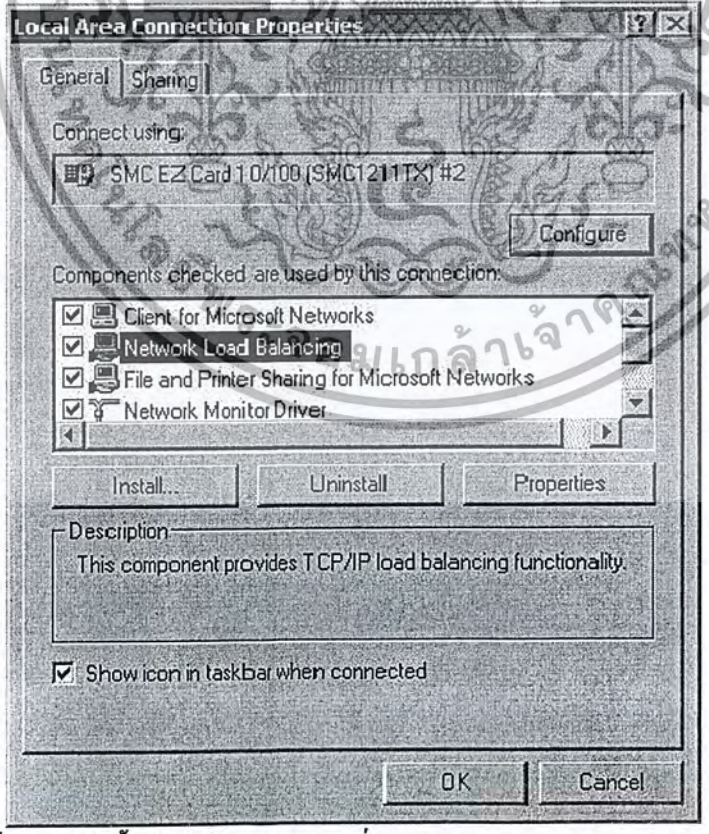
2. ทำการติดตั้ง IIS 5.0

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข-2 วิธีติดตั้ง IIS บนWindows 2000 Advanced Server

3. ทำการติดตั้ง Load Balancing ของ Windows 2000 Advanced Server

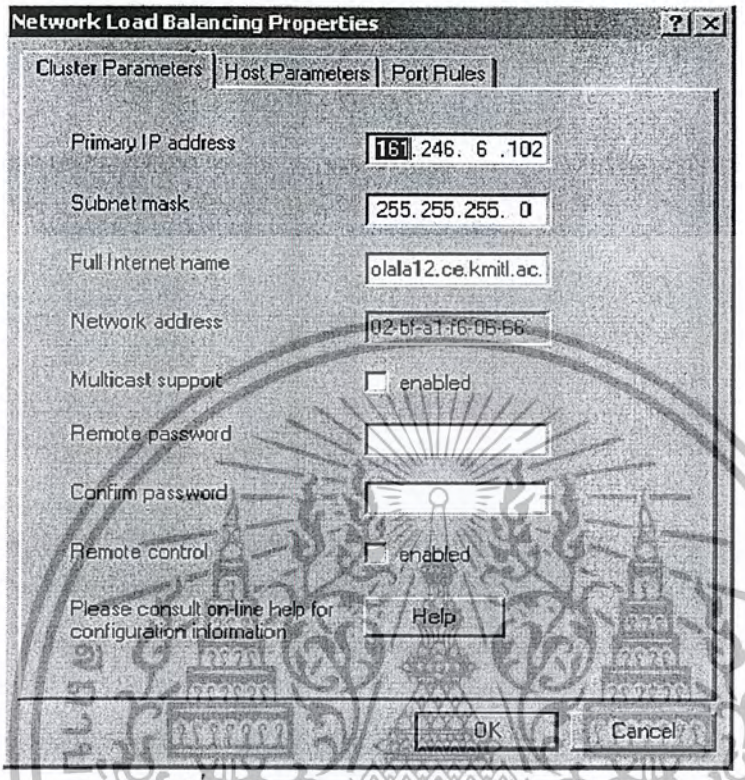


รูปที่ ข-3 วิธีติดตั้งเน็ตเวิร์คโหลดบาลานซิ่งบน Windows 2000 Advanced Server

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. กำหนดค่าต่างๆที่ใช้ใน Load Balancing

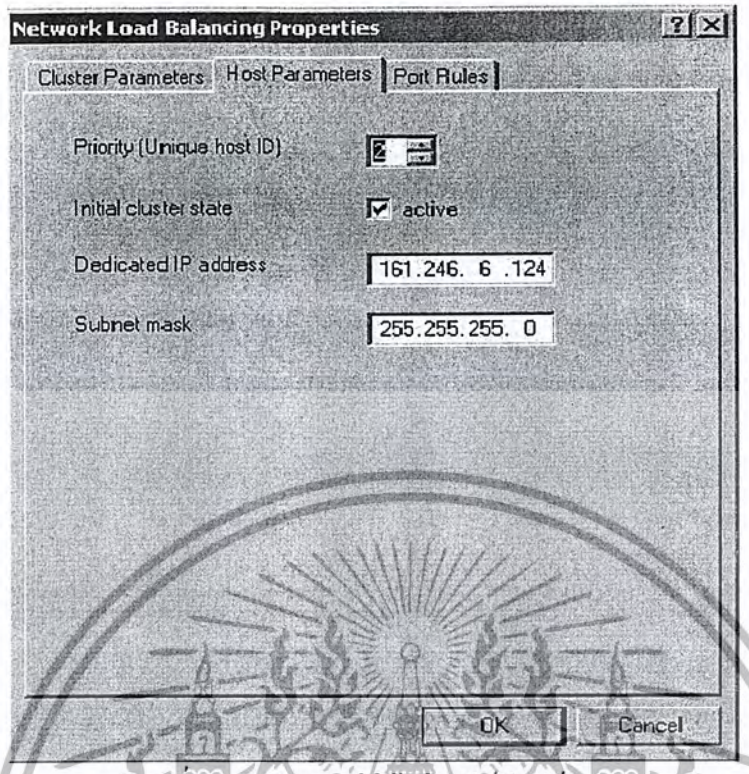
4.1. กำหนดให้ IP กลางเป็น 161.246.6.102



รูปที่ ข-4 การกำหนดค่าให้แต่ละเซิร์ฟเวอร์

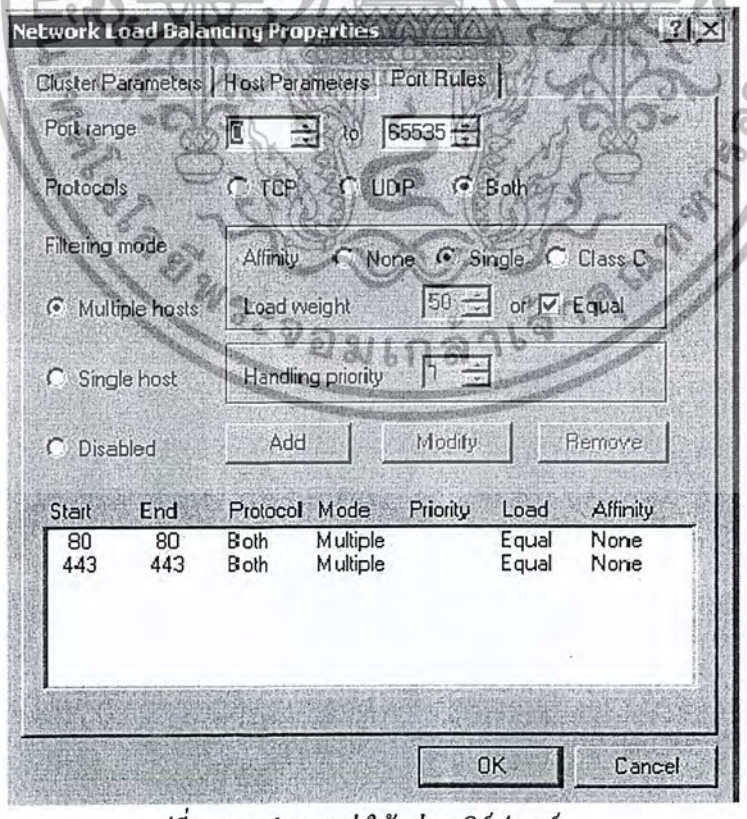
4.2. กำหนดค่าลำดับความสำคัญ (Priorities)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข-5 การกำหนดค่าให้แต่ละเซิร์ฟเวอร์

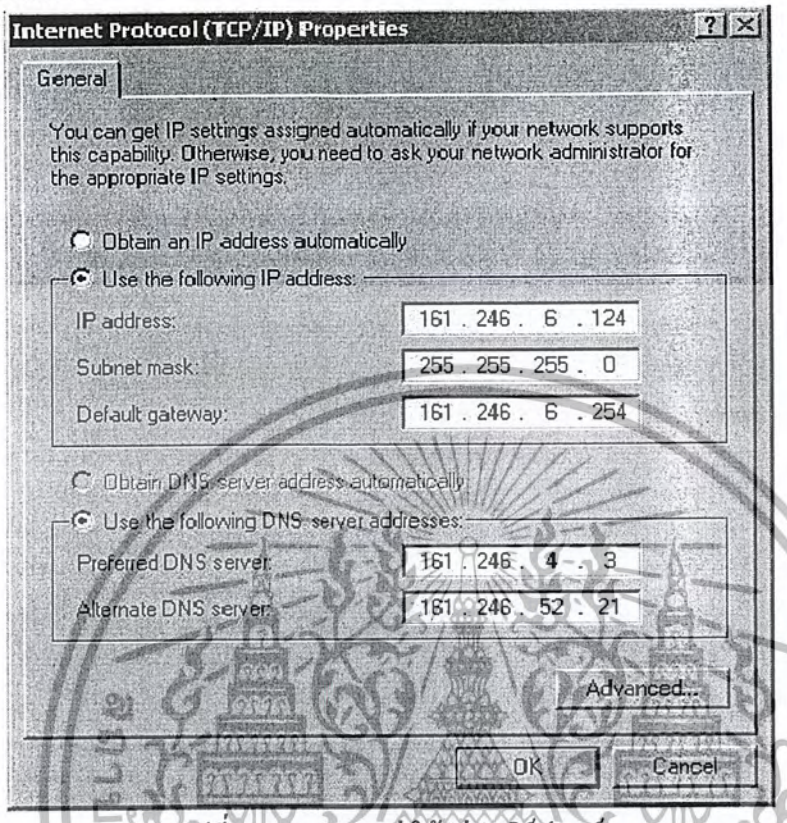
4.3. กำหนดว่าจะให้พอร์ตใดบ้างที่ใช้งาน Load Balancing



รูปที่ ข-6การกำหนดค่าให้แต่ละเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

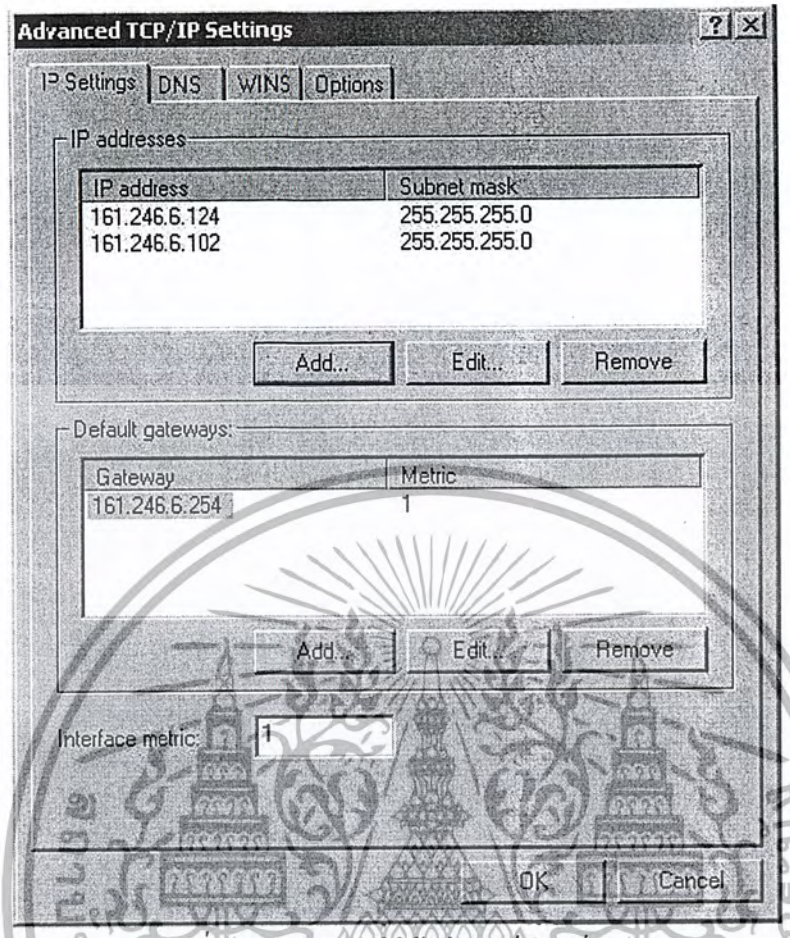
5. กำหนดไอพีให้เครื่อง



รูปที่ ข-7 การกำหนดค่าให้แต่ละเซิร์ฟเวอร์

6. เพิ่มอีกไอพีเพื่อทำ Load Balancing

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ ข-8 การกำหนดค่าให้แต่ละเซิร์ฟเวอร์

เซิร์ฟเวอร์ 2(Server 2)

การติดตั้งเหมือนกับเครื่องเซิร์ฟเวอร์ 1 ทุกอย่างเพียงแต่เปลี่ยนไอพีของเครื่องเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] Robert J. Shimonski : *"Windows Server 2003 Clustering & Load Balancing"* , Mc Graw Hill Osbourne ,2003

[2] BEA Systems , Inc : *"BEA WebLogic Server Administration Guide"* , 2001

[3] Ed Roman : *"Mastering Enterprise JavaBeans"* , Wiley Computer Publishing , 1999

[4] Ali Akbar,Keyur Shah : *"BEA Weblogic 7 Server Administration"* , Mc Graw Hill Osbourne, 2002

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้